Oracle® Database

Move to Oracle Cloud Using Zero Downtime Migration

Release 21c (21.1)
F38210-07
June 2022

Oracle Database Move to Oracle Cloud Using Zero Downtime Migration, Release 21c (21.1)

F38210-07

Contents

Preface

This book provides information about Zero Downtime Migration capabilities, how to set up the Zero Downtime Migration service, how to prepare your source and target databases for migration, and how to use the Zero Downtime Migration tool to quickly and smoothly move your Oracle databases to the Oracle Cloud or any Oracle Exadata Database Machine environment without incurring any significant downtime.

Audience

This book is intended for anyone who wants to learn about what Zero Downtime Migration can do, and for anyone tasked with migrating Oracle databases.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Related Documents

See Zero Downtime Migration on the Oracle Help Center for all published Zero Downtime Migration documentation.

See Zero Downtime Migration Release Notes for the latest information about what's new in this release, known issues, and troubleshooting solutions.

See the README file included with the downloaded Zero Downtime Migration software for additional information about installation.

See Zero Downtime Migration Licensing Information User Manual for information about third party licenses included in the Zero Downtime Migration software kit.

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1

Introduction to Zero Downtime Migration

Learn about how Zero Downtime Migration works, and its requirements and supported configurations.

About Zero Downtime Migration

With Zero Downtime Migration, you can migrate Oracle databases from on premises, Oracle Cloud Infrastructure Classic, or from one Oracle Cloud Infrastructure region to another. You can move your databases to co-managed or Autonomous Database services in the cloud, or any Exadata Database Machine in the cloud or on premises.

Zero Downtime Migration provides a robust, flexible, and resumable migration process that is also easy to roll back. Zero Downtime Migration integrates Oracle Maximum Availability Architecture (MAA) and supports Oracle Database 11g Release 2 (11.2.0.4) and later database releases.

You can perform and manage a database migration of an individual database or perform database migrations at a fleet level. Leveraging technologies such as Oracle Data Guard, Oracle Recovery Manager (RMAN), Oracle GoldenGate, and Oracle Data Pump, you can migrate databases online or offline.

The Zero Downtime Migration software is a service with a command line interface that you install and run on a host that you provision. The server where the Zero Downtime Migration software is installed is called the Zero Downtime Migration service host. You can run one or more database migration jobs from the Zero Downtime Migration service host.

Zero Downtime Migration Capabilities

- **Audit capability** - All custom user actions are audited including actions performed by the migration job.
- **Work flow customization** - You can customize the migration work flow (marked by operational phases) with your own scripts, which can be run before or after any phase in the work flow.
- **Job subsystem** - You can perform and manage database migrations at a fleet scale.
- **Job scheduler** - You can schedule your migration job to run at a future point in time.
- **Pause and resume functionality** - You can pause and resume your migration job if needed, which is useful to conform to a maintenance window, for example.
- **Job termination** - You can terminate a running migration job, rather than waiting for it to complete.
- **Job rerun ability** - Your migration job can be re-run (resumed) from a point of failure.
- **Job pre-check** - You can run pre-checks for migration tasks to prevent errors during database migration.
- **Compliance** - Zero Downtime Migration is compliant with Oracle Maximum Availability Architecture best practices and supports Oracle Database 11g Release 2 (11.2.0.4.0) and later.

Migration Methods

Zero Downtime Migration supports both online and offline migration, and can perform both physical and logical migrations. Consider the following advantages of each migration method to choose which one is right for your needs.

- **Online** migration methods incur zero or minimal downtime (typically less than 15 minutes) and can leverage either physical or logical migration methods.
- **Offline** migration methods will incur downtime on the source database as part of the migration process. It can leverage either physical or logical migration methods.

  Note that the only available method for migrating Oracle Database Standard Edition is the offline migration method.
- **Physical** migration methods:
  - Use Oracle Data Guard and RMAN to perform migrations
  - Allow you to convert a non-multitenant (non-CDB) source database to a multitenant (CDB) target database
- **Logical** migration methods:
  - Use Oracle Data Pump and, for online migrations, Oracle GoldenGate Microservices
  - Include integration with the Cloud Premigration Advisor Tool (CPAT), which a) warns you about any features used by your database that aren't supported in the target cloud environment, and b) makes suggestions for remedial changes and/or parameters to use for the Data Pump export and import operations

The migration methods are described in the following topics.

- Physical Migrations with Zero Downtime Migration
- Logical Migrations with Zero Downtime Migration

Physical Migrations with Zero Downtime Migration

Online and offline physical migrations using Zero Downtime Migration are described in the following topics:

Physical Online Migration

Zero Downtime Migration harnesses Oracle Data Guard to perform an online physical migration.

A Zero Downtime Migration online physical migration does the following:

- Backs up the source database to the specified data transfer medium
- Instantiates a standby database from this backup to the target environment
- Configures Data Guard with Maximum Performance protection mode and asynchronous (ASYNC) redo transport mode
- Synchronizes the source and target databases
- Switches over to the target database as the new primary database with minimum downtime

Upon switchover, the target database becomes the primary database, and the source database becomes the standby.

If there is SQL*Net connectivity between the new primary and the new standby after the switchover, the configuration continues to synchronize data (ship redo) from the new primary to the new standby source database. This configuration makes it possible to perform a rollback with minimal downtime, if you need to switch the primary back to the original source database.

However, if there is no SQL*Net connectivity between the new primary and the new standby after the switchover, there is no data synchronization (ship redo) from the new primary to the new standby on the source database. With this configuration you cannot fall back to the original source database.

Note that Transparent Data Encryption (TDE) is enabled on Oracle databases in the Oracle Cloud by default. Zero Downtime Migration handles the encryption of your target database, even if TDE is not enabled on the source Oracle database. However, once the switchover phase of the migration has taken place, the redo logs that the new primary database in the Oracle Cloud sends to the new standby database (the source) are encrypted. Therefore, if you decide to switch back and role swap again, making the source database the primary again and the database in the Oracle Cloud the standby, the source database will not be able to read the newly encrypted changed blocks applied by the redo logs unless TDE is enabled on the source database.

Physical Offline Migration

Zero Downtime Migration can perform a backup and restore operation to achieve an offline physical migration.

A Zero Downtime Migration offline physical migration does the following:

- Backs up the source database to the specified data transfer medium
- Instantiates a new database from this backup to the target environment

The offline migration method is similar to cloning a database. The target database has no relationship to the source, so there is no data synchronization or fallback capability. No SQL*Net connectivity is needed between the source and target database servers.

Supported Physical Migration Paths

Zero Downtime Migration supports a variety of physical migration paths.

- **On-Premises Database to Oracle Cloud Infrastructure**
  You can migrate an Oracle on-premises database to Oracle Cloud Infrastructure (either virtual machine or bare metal) with Zero Downtime Migration.

  In this scenario, Oracle Cloud Infrastructure Object Storage service is the supported intermediate data transfer medium for storing backups.
- **On-Premises Database to Oracle Exadata Cloud at Customer**
  You can migrate on-premises databases to Oracle Exadata Cloud at Customer environments with Zero Downtime Migration.

  In this scenario, Object Storage Service (OSS), Zero Data Loss Recovery Appliance (ZDLRA), or a Network File System (NFS) are the supported data transfer media for storing backups.
- **Oracle Cloud Infrastructure Classic Database to Oracle Cloud Infrastructure**

You can migrate a database in Oracle Cloud Infrastructure Classic to the Oracle Cloud Infrastructure (either virtual machine or bare metal) with Zero Downtime Migration.

In this scenario, Oracle Cloud Infrastructure Object Storage service is the supported intermediate data transfer medium for storing backups.

- **On-Premises Database to Exadata Cloud Service**
  You can migrate an Oracle on-premises database to Exadata Cloud Service with Zero Downtime Migration.

  In this scenario, Oracle Cloud Infrastructure Object Storage service is the supported intermediate data transfer medium for storing backups.
- **Oracle Cloud Infrastructure Database to Another Oracle Cloud Infrastructure Region**
  You can migrate a database from one Oracle Cloud Infrastructure region to another Oracle Cloud Infrastructure region with Zero Downtime Migration. For example, you can move a database from the `phoenix` region to the `frankfurt` or `ashburn` region.

  In this scenario, Oracle Cloud Infrastructure Object Storage service is the supported intermediate data transfer medium for storing backups.
- **On-Premises Database to On-Premises Exadata Database Machine**
  You can migrate on-premises databases to Oracle Exadata Database Machine with Zero Downtime Migration.

  In this scenario, Zero Data Loss Recovery Appliance (ZDLRA) or a Network File System (NFS) are the supported intermediate data transfer media for storing backups

Supported Data Transfer Media for Physical Migrations

Part of the Zero Downtime Migration process involves creating a backup of the source database and restoring it to the target database. Zero Downtime Migration supports Oracle Cloud Infrastructure Object Storage, Zero Data Loss Recovery Appliance, or NFS storage backup media, depending on your target environment.

- Object Storage Service (OSS)
- Zero Data Loss Recovery Appliance (ZDLRA)
- External Backup Location (NFS)

Oracle Cloud Infrastructure Object Storage

Object Storage is supported as a backup medium when migrating a database to Oracle Cloud Infrastructure, Exadata Cloud Service, or Exadata Cloud at Customer.

If you back up the database to Object Storage, then the Zero Downtime Migration service initiates the source database backup and restores it to the target environment, so Object Storage must be accessible from both the source and target environments.

The Zero Downtime Migration service host uses an SSH connection to the source and target database servers to install and configure the backup module software necessary to back up to and restore from Object Storage. The backup from the source database to Object Storage takes place over an RMAN channel.

Zero Data Loss Recovery Appliance

Zero Data Loss Recovery Appliance is supported as a backup medium when migrating a database to an Exadata Cloud at Customer target.

If Zero Data Loss Recovery Appliance is chosen as backup medium, then you must ensure that the Zero Data Loss Recovery Appliance has a valid backup of the source database, because Zero Downtime Migration does not initiate a backup to Zero Data Loss Recovery Appliance as part of the workflow.

You must also ensure that all instances of the database are up before initiating a backup to Zero Data Loss Recovery Appliance. The duplicate database operation might fail if the backup is initiated when an instance is down.

The Zero Downtime Migration service accesses the backup in Zero Data Loss Recovery Appliance and restores it to Exadata Cloud at Customer. The Zero Data Loss Recovery Appliance access credentials and wallet location are mandatory input parameters, so that Zero Downtime Migration can handle the Zero Data Loss Recovery Appliance wallet setup at the target database.

Any transfer of redo stream between the source and the target database server, in either direction, takes place over a SQL*Net link.

Refer to the Zero Data Loss Recovery Appliance documentation for information about creating backups.

Network File System (NFS)

NFS is supported as a backup medium when migrating a database to an Exadata Cloud at Customer target.

If you choose to back up the database to an NFS mount, then the Zero Downtime Migration service initiates the source database backup and restores it to the Exadata Cloud at Customer target environment. The NFS should be accessible from both the source and target environments.

Supported Database Architectures for Physical Migration

Zero Downtime Migration supports the following database architecture implementations.

- Oracle Database Single-Instance, which can be migrated to a single-instance or Oracle RAC database target
- Oracle RAC One Node, which can be migrated to an Oracle RAC database target
- Oracle RAC, which can be migrated to an Oracle RAC database target

Target Placeholder Database Environment

Zero Downtime Migration requires that you configure a placeholder database target environment before beginning the migration process. Zero Downtime Migration uses the provisioned target as a template and recreates the target during the course of migration.

You should configure the target database with the required and desired options of your native environment, because Zero Downtime Migration does not preserve this automatically during the migration.

During the migration process, the Zero Downtime Migration service host restores the source database to this placeholder database target environment by dropping the placeholder database and recreating a database in the target environment with the same `db_name` as that of source database.

Any database parameters for the target database, including SGA parameters, are maintained during the migration, and the migrated database runs with this same configuration.

Once the migration is complete, the target database is accessible using Oracle Database Cloud Service console, and you can manage the database with SRVCTL commands. You can make any modifications to database parameters after the migration.

Logical Migrations with Zero Downtime Migration

Online and offline logical migrations using Zero Downtime Migration are described in the following topics:

Logical Online Migration

Zero Downtime Migration harnesses Oracle GoldenGate and Oracle Data Pump to perform an online logical migration.

During a logical online migration, the source database remains online for client connections while data is moved to the target database, using a combination of Oracle Data Pump and Oracle GoldenGate replication.

The source database can be either a pluggable database (PDB) or a non-multitenant database.

Logical online migration involves two steps:

- Instantiation of target database.
  Oracle Data Pump extracts data from the source database and loads it into the target database.
- Real-time data replication between source and target databases.
  Oracle GoldenGate replicates data between the source and target databases in real-time until you choose to switch applications over to the target database.

Logical Offline Migration

Zero Downtime Migration can perform an offline logical migration using Oracle Data Pump to extract the data from the source database and load it into a target database.

Offline logical migration means that the source database is not available for clients while data is moved to the target database. When using the offline migration method, you must stop updates to the source database before you start a migration. When the migration is complete, the target database and source database do not require any direct SQL*Net connectivity between them.

Supported Logical Migration Paths

Zero Downtime Migration supports logical migration paths to a variety of target databases.

Based on your migration configuration, Zero Downtime Migration detects one of the following target database systems:

- **Autonomous Database Shared (Data Warehouse or Transaction Processing):**

  The following migration methods are supported for supported Autonomous Database targets:

  – Online or offline migration using a database link

  – Online or offline migration using OCI Object Storage

- **Autonomous Database Dedicated Infrastructure (Data Warehouse or Transaction Processing):**

  – Online or offline migration using OCI Object Storage

- **Co-managed Database Systems:** Exadata Cloud Service, Exadata Cloud at Customer, Virtual Machine, or Bare Metal

  The following migration methods are supported for supported co-managed database targets:

  – Online or offline migration using a database link

  – Online or offline migration using OCI Object Storage

  – Offline migration using NFS

  – Offline migration using Direct Copy

  – Offline migration using Object Storage with OCI CLI

Supported Data Transfer Media for Logical Migrations

Some Zero Downtime Migration logical migration paths involve placing Oracle Data Pump dump files on storage media for transfer to the target database.

For logical migrations, supported data transfer media are:

- **Oracle Cloud Object Store**

  Object Storage is supported as a Data Pump dump file storage medium for online and offline logical migrations to target Autonomous Database or co-managed databases.

- **Network File System (NFS)**

  In offline logical migrations to co-managed target databases, NFS is supported as a data transfer medium.

- **Direct Copy**

  Data Pump dumps are transferred directly from the source to target securely using secure copy (SCP) or RSYNC.

Data Replication

Replication migrates all data and metadata operations in transactions committed after the initial load until you resume the migration job after the Monitor Replication Lag phase. It includes inserts, deletes, and updates of data in tables within the migrated schema. Create, alter, and drop DDL operations are not replicated.

The following objects are not supported:

- Changes to external tables
- Oracle GoldenGate Unsupported Types (see Understanding What's Supported)

Zero Downtime Migration Requirements and Considerations

Supported Platforms

Zero Downtime Migration supports the following platforms for the service host and the migration source and target database servers.

Zero Downtime Migration Service Host - Supported Platforms

The Zero Downtime Migration service host can be configured on Oracle Linux 7 (Linux-x86-64) or later releases.

You can deploy the Zero Downtime Migration service on a standalone server on-premises or on a standalone Linux server (compute instance) in the Oracle Cloud. Oracle Linux is the supported platform for the Zero Downtime Migration service host.

Note that the Zero Downtime Migration service host can be shared with other applications for other purposes.

Source and Target Database Servers - Supported Platforms

Linux-x86-64 is the supported platform for co-managed source and target database servers.

Supported target systems are

- Oracle Cloud Infrastructure co-managed databases: Exadata Cloud Service, Exadata Cloud at Customer, Virtual Machine Database System, and Bare Metal Database System

  The target co-managed database can be either a pluggable database or a non-multitenant database.
- As a target environment only, Autonomous Database is supported platform for logical migrations. You can choose a Dedicated Infrastructure (Data Warehouse or Transaction Processing) or Shared (Data Warehouse or Transaction Processing).

  An Autonomous Database is a pluggable database in an Autonomous Container Database (ACD) deployed on an Autonomous Exadata Infrastructure (AEI) rack.
- On-premises Exadata Database Machine

Supported Database Versions for Migration

Zero Downtime Migration supports most Oracle Database versions available on Oracle Cloud Infrastructure, Exadata Cloud at Customer, and Exadata Cloud Service.

The following Oracle Database versions can be migrated using Zero Downtime Migration.

- Oracle Database 11g Release 2 (11.2.0.4)
- Oracle Database 12c Release 1 (12.1.0.2)
- Oracle Database 12c Release 2 (12.2.0.1)

- Oracle Database 18c
- Oracle Database 19c
- Oracle Database 21c
- All subsequent Oracle Database releases

> **Note:**
>
> The following notes apply only to physical migrations.
> Because Zero Downtime Migration physical migrations leverage Oracle Data Guard, you must have the same operating system and database version on both source and target. However, note that, while Standard Edition databases can use Zero Downtime Migration, they must use the offline migration method which is based on a backup and restore methodology and does not leverage Data Guard.
>
> Zero Downtime Migration physical migrations do not support cross-edition migration. Zero Downtime Migration cannot be used to migrate an Enterprise Edition database to a Standard Edition database, and vice versa.

Zero Downtime Migration Database Server Access

The Zero Downtime Migration service host needs to access the source and target database servers during a database migration.

To perform the migration, the Zero Downtime Migration service host requires either root user or SSH key-based access to one of the source database servers, and the Zero Downtime Migration service host requires SSH key-based access to one of the target database servers. If you are migrating an Oracle RAC database, providing access to one of the Oracle RAC nodes is adequate. The Zero Downtime Migration service host copies the software needed for migration to the source and target servers and cleans it up at the end of the operation.

An SSH private key is required to establish SSH connections. This generated key must not use a passphrase. You can create and add a new SSH key to your existing deployment using the Oracle Cloud Service Console.

Zero Downtime Migration Operational Phases

The Zero Downtime Migration service defines the migration process in units of operational phases.

Zero Downtime Migration auto computes the migration work flow using defined operational phases based on configured input parameters, such as the target platform, backup medium, and so on. You can customize the work flow by inserting custom plug-ins on each of the operational phases. The Zero Downtime Migration service lets you pause and resume the migration work flow at any chosen operational phase.

Migration work flow-associated phases for a given operation can be listed. Phases that are performed on the source database server are listed with a _SRC suffix, and the phases associated with the target database server are listed with a _TGT suffix.

See Also:

Zero Downtime Migration Process Phases

Zero Downtime Migration Security Provisions

Zero Downtime Migration permissions and ownership of files and directories, and handling of configurations for security features, are equivalent to those of Oracle Database.

Zero Downtime Migration installs in a location, named `ZDM_HOME`, that is structured similarly to the Oracle home directory, `ORACLE_HOME`, for Oracle Database. The permissions and ownership of files and directories in the `ZDM_HOME` follow the same conventions as that of a database `ORACLE_HOME`.

Zero Downtime Migration also creates a base directory structure for storing Zero Downtime Migration configuration files, logs, and other artifacts, named `ZDM_BASE`, that is similar to an Oracle base directory, `ORACLE_BASE`, that is associated with an Oracle home. The structure, owners, and permissions of directories and files in `ZDM_BASE` are similar to that of an `ORACLE_BASE`.

`ZDM_BASE` and `ORACLE_BASE` do not allow access by group or others.

You do not need to do any additional steps to ensure security the of the Zero Downtime Migration configuration because the Zero Downtime Migration configuration is designed to be secure out of the box.

Zero Downtime Migration is configured to accept JMX connections only from the local host, and to listen on the loopback address for HTTP connections. Zero Downtime Migration operations can only be performed by the operating system user that installed the product.

For physical migrations, SSH connectivity from the Zero Downtime Migration service host to the source database server and the target database server is required. You must provide the SSH key file location as an input for a migration job, and the existence of this file is expected for the duration of the migration job. You must manage the security of the directories and files where these key files are located.

You can modify the communication ports when there is a port conflict with another application. Note that access to these ports are configured only from within the Zero Downtime Migration host. You can change the RMI and HTTP port properties in the file `$ZDM_BASE/crsdata/`*`zdm_service_host`*`/rhp/conf/standalone_config.properties`.

The properties are:

- RMI port - `oracle.jwc.rmi.port=8895`
- HTTP port - `oracle.jwc.http.port=8896`

Restart the Zero Downtime Migration service after changing the properties.

When Zero Downtime Migration operations require passwords, prompts are given for password entry. Passwords are encrypted and stored in the Zero Downtime Migration database. Provided passwords are not expected to change for the duration of a migration job.

From an operation perspective, Zero Downtime Migration follows the guidelines in *Oracle Database Security Guide* for handling source and target database configurations for migration, such as Oracle Wallets, Transparent Data Encryption, and so on.

See Also:

Configuring Connectivity Prerequisites

Oracle Database Security Guide

2

Setting Up Zero Downtime Migration Software

When you install Zero Downtime Migration software, read this section carefully as there may have been changes since the last time you performed an installation.

The Zero Downtime Migration software kit supports both physical and logical migrations. You only need to install one kit to get all of the functionality.

Always see the Zero Downtime Migration Release Notes for the latest information about known issues. Also, see the README file included with the downloaded Zero Downtime Migration software for any additional information about software installation and updates.

For information about updating existing software to the latest release, removing the software, and starting and stopping the Zero Downtime Migration service, see Managing the Zero Downtime Migration Service.

Prepare a Host for Zero Downtime Migration Software Installation

If a host has not had Zero Downtime Migration software installed on it previously, verify that it complies with the requirements and perform any pre-installation tasks, then download and install the software. Once the software is installed, the host is referred to as the Zero Downtime Migration service host.

Provision a host with the following prerequisites and complete the following preinstallation tasks before installing Zero Downtime Migration software on it.

- The Zero Downtime Migration service host should be a dedicated system, but it can be shared for other purposes; however, the Zero Downtime Migration service host should not have Oracle Grid Infrastructure running on it.
- Zero Downtime Migration software requires a standalone Linux host running Oracle Linux 7.
- The Zero Downtime Migration service host must be able to connect to the source and the target database servers.
- Ensure that the Linux host has 100 GB of free storage space.
- You may use an existing user, or, on the Zero Downtime Migration service host, as root user, create a `zdm` group and add `zdmuser` user to the group.

  For example,

  ```
  root> groupadd zdm
  root> useradd -g zdm zdmuser
  ```

- Verify that the `glibc-devel` and `expect` packages are installed.

  For Oracle Linux 7 installations with Base Environment "Minimal Install" you also need to install the packages `unzip libaio oraclelinux-developer-release-el7`.

- Verify that the `/etc/hosts` entry for the host name and IP address are configured as expected, so that the host selected for Zero Downtime Migration software installation resolves to the correct IP address and the IP address is reachable with `ping`.
- During the installation, the script might report any missing packages and instructions for setting appropriate values for kernel parameters. Be sure to install the missing packages and set the kernel parameters before the Zero Downtime Migration software installation.
- Optionally, set a `ZDM_HOME` environment variable to the absolute path of the directory where the Zero Downtime Migration software will be installed. All of the examples in this document use `$ZDM_HOME`.

```
zdmuser> export ZDM_HOME=absolute_path_to_zdm_home
```

Install Zero Downtime Migration Software

All commands are run as `zdmuser`.

1. Download the Zero Downtime Migration software kit from https://www.oracle.com/database/technologies/rac/zdm-downloads.html to the Zero Downtime Migration service host.
2. Install the Zero Downtime Migration software as a non-root user.

   In this example the installation user is `zdmuser`.

   a. Change to the directory to where Zero Downtime Migration software is downloaded and unzip the software.

   ```
   zdmuser> cd zdm_download_directory
   zdmuser> unzip zdmversion.zip
   ```

   b. Run the Zero Downtime Migration installation script.

   ```
   zdmuser>./zdminstall.sh setup oraclehome=zdm_oracle_home
   oraclebase=zdm_base_directory
           ziploc=zdm_software_location -zdm
   ```

   - `zmdinstall.sh` is the installation script
   - `oraclehome` is the absolute path to the Oracle Home directory where the Zero Downtime Migration software will be installed.
   - `oraclebase` is the absolute path to the base directory where all of the Zero Downtime Migration configuration files, logs, and other artifacts are stored
   - `ziploc` is the location of the compressed software file (zip) included in the Zero Downtime Migration kit

   For example,

   ```
   zdmuser>./zdminstall.sh setup oraclehome=/u01/app/zdmhome
           oraclebase=/u01/app/zdmbase ziploc=/u01/app/oracle/zdm/
   ```

```
    shiphome/zdm_home.zip
            -zdm
```

Hereafter, the `oraclehome` value is referred to as `ZDM_HOME`, and the `oraclebase` value is referred to as `ZDM_BASE`.

Ignore the following messages which are displayed on the terminal at the end of installation. There is no need to run these scripts.

```
As a root user, execute the following script(s):
        1. $ZDM_HOME/inventory/orainstRoot.sh
        2. $ZDM_HOME/root.sh
```

3. Start the Zero Downtime Migration service as user `zdmuser`.

   ```
   zdmuser> $ZDM_HOME/bin/zdmservice start
   ```

   You must start `zdmservice` before you can migrate your databases using Zero Downtime Migration.

   If you must stop the Zero Downtime Migration service, run the following command.

   ```
   zdmuser> $ZDM_HOME/bin/zdmservice stop
   ```

4. Verify that the Zero Downtime Migration service installation is successful.

   When you run the following command, the output should be similar to that shown here.

   ```
   zdmuser> $ZDM_HOME/bin/zdmservice status
   ---------------------------------------
           Service Status
   ---------------------------------------
   Running:       true
   Tranferport:   5000-7000
   Conn String:   jdbc:mysql://localhost:8897/
   RMI port:      8895
   HTTP port:     8896
   Wallet path:   /u01/app/zdmbase/crsdata/fopds/security
   ```

5. If necessary, change the default MySQL port.

   Zero Downtime Migration uses MySQL internally, configuring it by default on port 8897, as shown in the above `zdmservice status` example output. If you want to change this port number, see Setting the MySQL Port.

3 Preparing for Database Migration

Before starting a Zero Downtime Migration database migration you must configure connectivity between the servers, prepare the source and target databases, set parameters in the response file, and configure any required migration job customization.

See the Zero Downtime Migration Release Notes for the latest information about new features, known issues, and My Oracle Support notes.

Preparing for a Physical Database Migration

The following topics describe how to configure the Zero Downtime Migration prerequisites before running a physical database migration job.

Configuring Connectivity Prerequisites

Connectivity must be set up between the Zero Downtime Migration service host and the source and target database servers.

The following topics describe how to configure the Zero Downtime Migration connectivity prerequisites before running a migration job.

Configuring Connectivity From the Zero Downtime Migration Service Host to the Source and Target Database Servers

Complete the following procedure to ensure the required connectivity between the Zero Downtime Migration service host and the source and target database servers.

1. On the Zero Downtime Migration service host, verify that the RSA authentication key pairs are available without a passphrase for the Zero Downtime Migration software installed user.

   If a new key pair must be generated without the passphrase, then, as a Zero Downtime Migration software installed user, generate new key pairs as described in Generate SSH Keys Without a Passphrase.

2. Rename the private key file.

   Rename the *zdm_installed_user_home*/.ssh/id_rsa file name to *zdm_installed_user_home*/.ssh/*zdm_service_host*.ppk.

3. Add the contents of the *zdm_installed_user_home*/.ssh/id_rsa.pub file to the *opc_user_home*/.ssh/authorized_keys file, with the following dependencies:

   For the source database server:

   - If the source database server is accessed with the root user, then no action is required.
   - If the source database server is accessed through SSH, then add the contents of the *zdm_installed_user_home*/.ssh/id_rsa.pub file into the *opc_user_home*/.ssh/authorized_keys file on all of the source database servers.

For the target database server:

- Because the target database server is on cloud only and access is through SSH, add the contents of the *zdm_installed_user_home*/.ssh/id_rsa.pub file into the *opc_user_home*/.ssh/authorized_keys file on *all* of the target database servers.

Note that the `opc` user is a standard Oracle cloud user that is used to access database servers, but you can use any privileged user that has sudo privileges. You can also use different users for the source and target databases.

4. Make sure that the source and target database server names are resolvable from the Zero Downtime Migration service host through either resolving name servers or alternate ways approved by your IT infrastructure.

   One method of resolving source and target database server names is to add the source and target database server names and IP address details to the Zero Downtime Migration service host `/etc/hosts` file.

   In the following example, the IP address entries are shown as 192.x.x.x, but you must add your actual public IP addresses.

   ```
   #OCI public IP two node RAC server details
   192.0.2.1 ocidb1
   192.0.2.2 ocidb2
   #OCIC public IP two node RAC server details
   192.0.2.3 ocicdb1
   192.0.2.4 ocicdb2
   ```

   Optionally, Zero Downtime Migration allows connectivity through bastion hosts for both logical and physical migrations.

5. Make certain that port 22 in the source and target database servers accept incoming connections from the Zero Downtime Migration service host.

6. Test the connectivity from the Zero Downtime Migration service host to all source and target database servers.

   ```
   zdmuser> ssh -i zdm_service_host_private_key_file_location
   user@source/target_database_server_name
   ```

   For example,

   ```
   zdmuser> ssh -i /home/zdmuser/.ssh/zdm_service_host.ppk opc@ocidb1
   zdmuser> ssh -i /home/zdmuser/.ssh/zdm_service_host.ppk opc@ocicdb1
   ```

   Note:

   SSH connectivity during Zero Downtime Migration operations requires direct, non-interactive access between the Zero Downtime Migration service host and the source and target database servers without the need to enter a passphrase.

7. Disable TTY and verify that it is disabled for the SSH privileged user.

TTY needs to be turned off so that Zero Downtime Migration can run commands on the remote hosts non-interactively.

Because there are many ways to set sudo privileges, there are many ways to disable TTY for the `zdmuser`. As an example, you could set the following default in `/etc/sudoers` file.

```
Defaults:zdmuser !requiretty
```

Run the following command to verify that TTY is disabled:

```
ssh -i zdm_service_host_private_key_file_location
 user@source_database/target_database_server_name
 "sudo_location_source/target_database /bin/sh -c date"
```

If TTY is disabled, the command above returns the date from the remote host without any errors.

If SSH is configured to require TTY, the output shows an error, such as the following:

```
[opc@zdm-server ~]$ ssh -i /home/zdmuser/.ssh/zdm_service_host.ppk
opc@ocidb1
 "/usr/bin/sudo /bin/sh -c date"

sudo: sorry, you must have a tty to run sudo
```

See Also:

Zero Downtime Migration Port Requirements

Configuring SUDO Access

You may need to grant certain users authority to perform operations using `sudo` on the source and target database servers.

For source database servers:

- If the source database server is accessed with the `root` user, then there is no need to configure Sudo operations.
- If the source database server is accessed through SSH, then configure Sudo operations to run without prompting for a password for the database installed user and the `root` user.

  For example, if database installed user is `oracle`, then run `sudo su - oracle`.

  For the `root` user run `sudo su -`.

For target database servers:

- Because the target database server is on the cloud only, any Sudo operations are configured already. Otherwise, configure all Sudo operations to run without prompting for a password for the database installed user and the `root` user.

  For example, if database installed user is `oracle`, then run `sudo su - oracle`.

  For the `root` user run `sudo su -`.

Note, for example, if the login user is `opc`, then you can enable Sudo operations for the `opc` user.

Configuring Connectivity Between the Source and Target Database Servers

You have two options for configuring connectivity between the source and target database servers: SQL*Net connectivity using SCAN or SSH.

Configure connectivity using one of the following options.

Option 1: SQL*Net Connectivity Using SCAN

To use this option, the SCAN of the target should be resolvable from the source database server, and the SCAN of the source should be resolvable from the target server.

The specified source database server in the `ZDMCLI migrate database` command `-sourcenode` parameter can connect to the target database instance over target SCAN through the respective SCAN port and vice versa.

With SCAN connectivity from both sides, the source database and target databases can synchronize from either direction. If the source database server SCAN cannot be resolved from the target database server, then the `SKIP_FALLBACK` parameter in the response file must be set to `TRUE`, and the target database and source database cannot synchronize after switchover.

Test Connectivity

To test connectivity from the source to the target environment, add the TNS entry of the target database to the source database server `$ORACLE_HOME/network/admin/tnsnames.ora` file.

```
[oracle@sourcedb ~] tnsping target-tns-string
```

To test connectivity from the target to the source environment, add the TNS entry of the source database to the target database server `$ORACLE_HOME/network/admin/tnsnames.ora` file

```
[oracle@targetdb ~] tnsping source-tns-string
```

Note:

Database migration to Exadata Cloud at Customer using the Zero Data Loss Recovery Appliance requires mandatory SQL*Net connectivity from the target database server to the source database server.

See Also:

Zero Downtime Migration Port Requirements

Option 2: Set up an SSH Tunnel

If connectivity using SCAN and the SCAN port is not possible between the source and target database servers, set up an SSH tunnel from the source database server to the target database server.

The following procedure sets up an SSH tunnel on the source database servers for the root user. Note that this procedure amounts to setting up what may be considered a temporary channel. Using this connectivity option, you will not be able to synchronize between the target database and source database after switchover, and with this configuration you cannot fall back to the original source database.

Note:

The following steps refer to Oracle Cloud Infrastructure, but are also applicable to Exadata Cloud at Customer and Exadata Cloud Service.

1. Generate an SSH key file without a passphrase for the `opc` user on the target Oracle Cloud Infrastructure server, using the information in Generate SSH Keys Without a Passphrase. If the target is an Oracle RAC database, then generate an SSH key file without a passphrase from the first Oracle RAC server.
2. Add the contents of the Oracle Cloud Infrastructure server opc_user_home/.ssh/id_rsa.pub file into the Oracle Cloud Infrastructure server opc_user_home/.ssh/authorized_keys file.
3. Copy the target Oracle Cloud Infrastructure server private SSH key file onto the source server in the /root/.ssh/ directory. If the source is an Oracle RAC database, copy the file into all of the source servers.

   For better manageability, keep the private SSH key file name the same as the target server name, and keep the .ppk extension. For example, ocidb1.ppk (where ocidb1 is the target server name).
   The file permissions should be similar to the following.

   ```
   /root/.ssh>ls -l ocidb1.ppk
   -rw------- 1 root root 1679 Oct 16 10:05 ocidb1.ppk
   ```

4. Put the following entries in the source server /root/.ssh/config file.

   ```
   Host *
     ServerAliveInterval 10
     ServerAliveCountMax 2

   Host OCI_server_name
     HostName OCI_server_IP_address
     IdentityFile Private_key_file_location
   ```

```
  User OCI_user_login
  ProxyCommand /usr/bin/nc -X connect -x proxy_name:proxy_port %h %p
```

Where

- *OCI_server_name* is the Oracle Cloud Infrastructure target database server name without the domain name. For an Oracle RAC database use the first Oracle RAC server name without the domain name.
- *OCI_server_IP_address* is the Oracle Cloud Infrastructure target database server IP address. For an Oracle RAC database use the first Oracle RAC server IP address.
- *Private_key_file_location* is the location of the private key file on the source database server, which you copied from the target database server in step 3 above.
- *OCI_user_login* is the OS user used to access the target database servers.
- *proxy_name* is the host name of the proxy server.
- *proxy_port* is the port of the proxy server.

Note that the proxy setup might not be required when you are not using a proxy server for connectivity. For example, when the source database server is on Oracle Cloud Infrastructure Classic, you can remove or comment the line starting with ProxyCommand.

For example, after specifying the relevant values, the /root/.ssh/config file should be similar to the following.

```
Host *
  ServerAliveInterval 10
  ServerAliveCountMax 2

Host ocidb1
  HostName 192.0.2.1
  IdentityFile /root/.ssh/ocidb1.ppk
  User opc
  ProxyCommand /usr/bin/nc -X connect -x www-proxy.example.com:80
%h %p
```

The file permissions should be similar to the following.

```
/root/.ssh>ls -l config
-rw------- 1 root root 1679 Oct 16 10:05 config
```

In the above example, the Oracle Cloud Infrastructure server name is ocidb1, and the Oracle Cloud Infrastructure server public IP address is 192.0.2.1.

If the source is an Oracle Cloud Infrastructure Classic server, the *proxy_name* is not required, so you can remove or comment the line starting with `ProxyCommand`.

If the source is an Oracle RAC database, then copy the same /root/.ssh/config file onto all of the source Oracle RAC database servers. This file will have the Oracle Cloud Infrastructure server name, Oracle Cloud Infrastructure server public IP address, and private key file location of first Oracle Cloud Infrastructure Oracle RAC server information configured.

5. Make sure that you can SSH to the first target Oracle Cloud Infrastructure server from the source server before you enable the SSH tunnel.

   For an Oracle RAC database, test the connection from all of the source servers to the first target Oracle Cloud Interface server.
   Using the private key:

   ```
   [root@ocicdb1 ~] ssh -i /root/.ssh/ocidb1.ppk opc@ocidb1
   Last login: Fri Dec  7 14:53:09 2018 from 192.0.2.3

   [opc@ocidb1 ~]$
   ```

   > **Note:**
   >
   > SSH connectivity requires direct, non-interactive access between the source and target database servers, without the need to enter a passphrase.

6. Run the following command on the source server to enable the SSH tunnel.

   ```
   ssh -f OCI_hostname_without_domain_name -L
   ssh_tunnel_port_number:OCI_server_IP_address:OCI_server_listener_port -N
   ```

   Where

   - *OCI_hostname_without_domain_name* is the Oracle Cloud Infrastructure target database server name without a domain name. For an Oracle RAC database use the first Oracle RAC server name without domain name.
   - *ssh_tunnel_port_number* is any available ephemeral port in the range (1024-65545). Make sure that the SSH tunnel port is not used by any other process in the server before using it.
   - *OCI_server_listener_port* is the target database listener port number. The listener port must be open between the source database servers and Oracle Cloud Infrastructure target servers.
   - *OCI_server_IP_address* is the IP address of the target database server. For a single instance database, specify the Oracle Cloud Infrastructure server IP address. For an Oracle RAC database, specify the Oracle Cloud Infrastructure scan name with the domain name. If the scan name with domain name is not resolvable or not working, then specify the IP address obtained using the `lsnrctl status` command output. For example,

     ```
     Listening Endpoints Summary...
       (DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(KEY=LISTENER)))
       (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=192.0.2.9)(PORT=1521)))
       (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=192.0.2.10)(PORT=1521)))
     ```

   The following is an example of the command run to enable the SSH tunnel.

   ```
   [root@ocicdb1~]ssh -f ocidb1 -L 9000:192.0.2.9:1521 -N
   ```

   For an Oracle RAC database, this step must be repeated on all of the source servers.

7. Test the SSH tunnel.

   Log in to source server, switch to the `oracle` user and source the database environment, and run the following command.

   ```
   tnsping localhost:ssh_tunnel_port
   ```

   For example,

   ```
   [oracle@ocicdb1 ~] tnsping localhost:9000
   ```

   The command output is similar to the following.

   ```
   TNS Ping Utility for Linux: Version 12.1.0.2.0 - Production on 22-
   JAN-2019 05:41:57
   Copyright (c) 1997, 2014, Oracle.  All rights reserved.
   Used parameter files:
   Used HOSTNAME adapter to resolve the alias
   Attempting to contact (DESCRIPTION=(CONNECT_DATA=(SERVICE_NAME=))
   (ADDRESS=(PROTOCOL=TCP)(HOST=127.0.0.1)(PORT=9000)))
   OK (50 msec)
   ```

   If tnsping does not work, then the SSH tunnel is not enabled.

   For Oracle RAC, this step must be repeated on all of the source servers.

Generate SSH Keys Without a Passphrase

You can generate a new SSH key without a passphrase if on the Zero Downtime Migration service host the authentication key pairs are not available without a passphrase for the Zero Downtime Migration software installed user.

Note:

Currently, only the RSA key format is supported for configuring SSH connectivity, so use the `ssh-keygen` command, which generates both of the authentication key pairs (public and private).

The following example shows you how to generate an SSH key pair for the Zero Downtime Migration software installed user. You can also use this command to generate the SSH key pair for the `opc` user.

Run the following command on the Zero Downtime Migration service host.

```
zdmuser> ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/zdmuser/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/zdmuser/.ssh/id_rsa.
Your public key has been saved in /home/zdmuser/.ssh/id_rsa.pub.
```

```
The key fingerprint is:
c7:ed:fa:2c:5b:bb:91:4b:73:93:c1:33:3f:23:3b:30 zdmuser@zdm_service_host
The key's randomart image is:
+--[ RSA 2048]----+
|                 |
|                 |
|                 |
|         . . .   |
|        S o . =  |
|         . E . * |
|           X.+o.|
|          .= Bo.o|
|          o+*o.  |
+-----------------+
```

This command generates the `id_rsa` and `id_rsa.pub` files in the `zdmuser` home, for example, `/home/zdmuser/.ssh`.

Preparing the Source and Target Databases

There are several tasks you must complete on the source and target databases before configuring a migration job.

Source Database Prerequisites

Meet the following prerequisites on the source database before the Zero Downtime Migration process starts.

- The source database must be running in `ARCHIVELOG` mode. See Changing the Database Archiving Mode.
- The character set on the source database must be the same as the target database.
- Configure the TDE wallet on Oracle Database 12c Release 2 and later. Enabling TDE on Oracle Database 11g Release 2 (11.2.0.4) and Oracle Database 12c Release 1 is optional.

  For Oracle Database 12c Release 2 and later, if the source database does not have Transparent Data Encryption (TDE) enabled, then it is mandatory that you configure the TDE wallet before migration begins. The `WALLET_TYPE` can be `AUTOLOGIN` (preferred) or `PASSWORD` based.

  Ensure that the wallet `STATUS` is `OPEN` and `WALLET_TYPE` is `AUTOLOGIN` (For an `AUTOLOGIN` wallet type), or `WALLET_TYPE` is `PASSWORD` (For a `PASSWORD` based wallet type). For a multitenant database, ensure that the wallet is open on all PDBs as well as the CDB, and the master key is set for all PDBs and the CDB.

  ```
  SQL> SELECT * FROM v$encryption_wallet;
  ```

- If the source is an Oracle RAC database, and `SNAPSHOT CONTROLFILE` is not on a shared location, configure `SNAPSHOT CONTROLFILE` to point to a shared location on all Oracle RAC nodes to avoid the ORA-00245 error during backups to Oracle Object Store.

For example, if the database is deployed on ASM storage,

```
$ rman target /
RMAN> CONFIGURE SNAPSHOT CONTROLFILE NAME TO '+DATA/db_name/
snapcf_db_name.f';
```

If the database is deployed on an ACFS file system, specify the shared ACFS location in the above command.

- Verify that port 22 on the source database server allows incoming connections from the Zero Downtime Migration service host.
- Ensure that the scan listener ports (1521, for example) on the source database servers allow incoming connections from the target database servers and outgoing connections to the target database servers.

  Alternate SQL connectivity should be made available if a firewall blocks incoming remote connection using the SCAN listener port.
- To preserve the source database Recovery Time Objective (RTO) and Recovery Point Objective (RPO) during the migration, the existing RMAN backup strategy should be maintained.

  During the migration a dual backup strategy will be in place; the existing backup strategy and the strategy used by Zero Downtime Migration. Avoid having two RMAN backup jobs running simultaneously (the existing one and the one initiated by Zero Downtime Migration). If archive logs were to be deleted on the source database, and these archive logs are needed by Zero Downtime Migration to synchronize the target cloud database, then these files should be restored so that Zero Downtime Migration can continue the migration process.
- If the source database is deployed using Oracle Grid Infrastructure and the database is not registered using SRVCTL, then you must register the database before the migration.
- The source database must use a server parameter file (SPFILE).
- If RMAN is not already configured to automatically back up the control file and SPFILE, then set `CONFIGURE CONTROLFILE AUTOBACKUP` to `ON` and revert the setting back to `OFF` after migration is complete.

  ```
  RMAN> CONFIGURE CONTROLFILE AUTOBACKUP ON;
  ```
- For **offline** migrations, plan to make sure no incoming transactions take place on the source database before the `ZDM_BACKUP_DIFFERENTIAL_SRC` phase, so that there is no loss of data during the migration. Once Zero Downtime Migration starts generating backups and transfers them, any new transactions on the source won't be part of the backups and therefore the target in the cloud won't have those changes.
- System time of the Zero Downtime Migration service host and source database server should be in sync with your Oracle Cloud Infrastructure target.

  If the time on any of these systems varies beyond 6 minutes from the time on OCI, it should be adjusted. You can use `ntp time check` to synchronize the time if NTP is configured. If NTP is not configured, then it is recommended that you configure it. If configuring NTP is not an option, then you need to correct the time manually to ensure it is in sync with OCI time.

- Set the `COMPATIBLE` database initialization parameter to the same value on the source and target database. See Values for the COMPATIBLE Initialization Parameter in Oracle Database for valid values.

> **See Also:**
>
> Setting Up the Transparent Data Encryption Keystore
> Zero Downtime Migration Port Requirements

Target Database Prerequisites

The following prerequisites must be met on the target database before you begin the Zero Downtime Migration process.

- You must create a placeholder target database.

  For Exadata Cloud Service and Exadata Cloud at Customer targets, the placeholder database must be created using Control Plane, not Grid Infrastructure Database Services before database migration begins.

  > **Note:**
  >
  > For this release, only Grid Infrastructure-based database services are supported as targets. For example, an LVM-based instance or an instance created in compute node without Grid Infrastructure are not supported targets.

  The placeholder target database is overwritten during migration, but it retains the overall configuration.

  Pay careful attention to the following requirements:

  – **Size for the future** - When you create the database from the console, ensure that your chosen shape can accommodate the source database, plus any future sizing requirements. A good guideline is to use a shape similar to or larger in size than source database.

  – **Set name parameters**

    * `DB_NAME` - If the target database is Exadata Cloud Service or Exadata Cloud at Customer, then the database `DB_NAME` should be the same as the source database `DB_NAME`. If the target database is Oracle Cloud Infrastructure, then the database `DB_NAME` can be the same as or different from the source database `DB_NAME`.

    * `DB_UNIQUE_NAME`: If the target database is Oracle Cloud Infrastructure, Exadata Cloud Service, or Exadata Cloud at Customer, the target database `DB_UNIQUE_NAME` parameter value must be unique to ensure that Oracle Data Guard can identify the target as a different database from the source database.

  – **Match the source SYS password** - Specify a `SYS` password that matches that of the source database.

  – **Disable automatic backups** - Provision the target database from the console without enabling automatic backups.

For Oracle Cloud Infrastructure and Exadata Cloud Service, do not select the **Enable automatic backups** option under the section **Configure database backups**.

For Exadata Cloud at Customer, set Backup destination **Type** to `None` under the section **Configure Backups**.

- The target database version should be the same as the source database version. The target database patch level should also be the same as (or higher than) the source database.

  If the target database environment is at a higher patch level than the source database (for example, if the source database is at Jan 2020 PSU/BP and the target database is at April 2020 PSU/BP), then Zero Downtime Migration runs the datapatch utility as part of the migration.
- The character set on the source database must be the same as the target database.
- The target database time zone version must be the same as the source database time zone version. To check the current time zone version, query the `V$TIMEZONE_FILE` view as shown here, and upgrade the time zone file if necessary.

  ```
  SQL> SELECT * FROM v$timezone_file;
  ```
- Verify that the TDE wallet folder exists, and ensure that the wallet `STATUS` is `OPEN` and `WALLET_TYPE` is `AUTOLOGIN` (For an auto-login wallet type), or `WALLET_TYPE` is `PASSWORD` (For a password-based wallet). For a multitenant database, ensure that the wallet is open on all PDBs as well as the CDB, and the master key is set for all PDBs and the CDB.

  ```
  SQL> SELECT * FROM v$encryption_wallet;
  ```
- The target database must use a server parameter file (SPFILE).
- If the target is an Oracle RAC database, then verify that SSH connectivity without a passphrase is set up between the Oracle RAC servers for the oracle user.
- Check the size of the disk groups and usage on the target database (ASM disk groups or ACFS file systems) and make sure adequate storage is provisioned and available on the target database servers.
- Make sure adequate storage is provisioned and available on the object store to accommodate the source database backup.
- Verify that ports 22 and 1521 (or the configured database listener port) on the target servers in the Oracle Cloud Infrastructure, Exadata Cloud Service, or Exadata Cloud at Customer environment are open and not blocked by a firewall.
- Verify that port 22 on the target database server allows incoming connections from the Zero Downtime Migration service host.
- Capture the output of the RMAN `SHOW ALL` command, so that you can compare RMAN settings after the migration, then reset any changed RMAN configuration settings to ensure that the backup works without any issues.

  ```
  RMAN> show all;
  ```

- System time of the Zero Downtime Migration service host and source database server should be in sync with your Oracle Cloud Infrastructure target.

  If the time on any of these systems varies beyond 6 minutes from the time on OCI, it should be adjusted. You can use `ntp time check` to synchronize the time if NTP is configured. If NTP is not configured, then it is recommended that you configure it. If configuring NTP is not an option, then you need to correct the time manually to ensure it is in sync with OCI time.
- Set the `COMPATIBLE` database initialization parameter to the same value on the source and target database. See Values for the COMPATIBLE Initialization Parameter in Oracle Database for valid values.

> **See Also:**
>
> Managing User Credentials for information about generating the auth token for Object Storage backups
>
> Zero Downtime Migration Port Requirements

Setting Up the Transparent Data Encryption Keystore

For Oracle Database 12c Release 2 and later, if the source and target databases do not have Transparent Data Encryption (TDE) enabled, then it is mandatory that you configure the TDE keystore before migration begins.

TDE should be enabled and the `TDE WALLET` status on both source and target databases must be set to `OPEN`. The `WALLET_TYPE` can be `AUTOLOGIN`, for an auto-login keystore (preferred), or `PASSWORD`, for a password-based keystore. On a multitenant database, make sure that the keystore is open on all PDBs as well as the CDB, and that the master key is set for all PDBs and the CDB.

If TDE is not already configured as required on the source and target databases, use the following instructions to set up the TDE keystore.

For a password-based keystore, you only need to do steps 1, 2, and 4; for an auto-login keystore, complete all of the steps.

1. Set `ENCRYPTION_WALLET_LOCATION` in the `$ORACLE_HOME/network/admin/sqlnet.ora` file.

   ```
   /home/oracle>cat /u01/app/oracle/product/12.2.0.1/dbhome_4/network/admin/
   sqlnet.ora

   ENCRYPTION_WALLET_LOCATION=(SOURCE=(METHOD=FILE)
     (METHOD_DATA=(DIRECTORY=/u01/app/oracle/product/12.2.0.1/dbhome_4/
   network/admin/)))
   ```

   For an Oracle RAC instance, also set `ENCRYPTION_WALLET_LOCATION` in the second Oracle RAC node.
2. Create and configure the keystore.

a. Connect to the database and create the keystore.

```
$ sqlplus "/as sysdba"
SQL> ADMINISTER KEY MANAGEMENT CREATE KEYSTORE '/u01/app/oracle/
product/12.2.0.1/dbhome_2/network/admin'
 identified by password;
```

b. Open the keystore.

   For a non-CDB environment, run the following command.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY
password;
keystore altered.
```

   For a CDB environment, run the following command.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY
password container = ALL;
keystore altered.
```

c. Create and activate the master encryption key.

   For a non-CDB environment, run the following command.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEY IDENTIFIED BY password
with backup;
keystore altered.
```

   For a CDB environment, run the following command.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEY IDENTIFIED BY password
with backup container = ALL;
keystore altered.
```

d. Query `V$ENCRYPTION_KEYS` to get the keystore status, keystore type, and keystore location.

```
SQL> SELECT * FROM v$encryption_keys;

WRL_TYPE    WRL_PARAMETER
--------------------
-----------------------------------------------------------------
---------------
STATUS                         WALLET_TYPE          WALLET_OR
FULLY_BAC     CON_ID
------------------------------ -------------------- ---------
---------            ----------
FILE        /u01/app/oracle/product/12.2.0.1/dbhome_2/network/
admin/
OPEN                           PASSWORD             SINGLE
NO         0
```

The configuration of a password-based keystore is complete at this stage, and the keystore is enabled with status `OPEN` and `WALLET_TYPE` is shown as `PASSWORD` in the query output above.

Continue to step 3 only if you need to configure an auto-login keystore, otherwise skip to step 4.

3. For an auto-login keystore only, complete the keystore configuration.

   a. Create the auto-login keystore.

   ```
   SQL> ADMINISTER KEY MANAGEMENT CREATE AUTO_LOGIN KEYSTORE FROM
   KEYSTORE
    '/u01/app/oracle/product/12.2.0.1/dbhome_2/network/admin/'
   IDENTIFIED BY password;
   keystore altered.
   ```

   b. Close the password-based keystore.

   ```
   SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE CLOSE IDENTIFIED BY
   password;
   keystore altered.
   ```

   c. Query `V$ENCRYPTION_WALLET` to get the keystore status, keystore type, and keystore location.

   ```
   SQL> SELECT * FROM v$encryption_wallet;
   WRL_TYPE WRL_PARAMETER
   --------------------
   ----------------------------------------------------------------------
   ----------
   STATUS WALLET_TYPE WALLET_OR FULLY_BAC CON_ID
   ------------------------------ -------------------- ---------
   --------- ---------
   FILE /u01/app/oracle/product/12.2.0.1/dbhome_2/network/admin/
   OPEN AUTOLOGIN SINGLE NO
   ```

   In the query output, verify that the TDE keystore `STATUS` is `OPEN` and `WALLET_TYPE` set to `AUTOLOGIN`, otherwise the auto-login keystore is not set up correctly.

   This completes the auto-login keystore configuration.

4. Copy the keystore files to the second Oracle RAC node.

   If you configured the keystore in a shared file system for Oracle RAC, or if you are enabling TDE for a single instance database, then no action is required.

   If you are enabling TDE for Oracle RAC database without shared access to the keystore, copy the following files to the same location on second node.

   - `/u01/app/oracle/product/12.2.0.1/dbhome_2/network/admin/ew*`
   - `/u01/app/oracle/product/12.2.0.1/dbhome_2/network/admin/cw*`

Preparing the Physical Migration Response File

Set the required physical migration response file parameters. Get the response file template, `$ZDM_HOME/rhp/zdm/template/zdm_template.rsp`, which is used to create your

Zero Downtime Migration response file for the database migration procedure, and edit the file as described here.

The following response file settings show you how to configure a typical use case. To further customize your configuration you can find additional parameters described in Zero Downtime Migration Physical Migration Response File Parameters Reference.

TGT_DB_UNIQUE_NAME

Set `TGT_DB_UNIQUE_NAME` to the target database `DB_UNIQUE_NAME` value. To find `DB_UNIQUE_NAME` run

```
SQL> show parameter db_unique_name
```

For Cloud type Exadata Cloud at Customer Gen 1, set `TGT_DB_UNIQUE_NAME` to a different `DB_UNIQUE_NAME` not currently in use.

PLATFORM_TYPE

Set `PLATFORM_TYPE` to one of the following:

- `VMDB` - Oracle Cloud Infrastructure virtual machine or bare metal targets.
- `EXACS` - Exadata Cloud Service
- `EXACC` - Exadata Cloud at Customer
- `NON_CLOUD` - On-premises Exadata Database Machine

MIGRATION_METHOD

Set `MIGRATION_METHOD` to one of the following:

- `ONLINE_PHYSICAL` - Oracle Data Guard (online)
- `OFFLINE_PHYSICAL` - RMAN backup and restore (offline). Note that this is the only migration method supported for Oracle Standard Edition databases.

DATA_TRANSFER_MEDIUM

`DATA_TRANSFER_MEDIUM` specifies the media used for the source database backup. Valid values are dependent on whether `MIGRATION_METHOD=ONLINE_PHYSICAL` or `MIGRATION_METHOD=OFFLINE_PHYSICAL`.

When `MIGRATION_METHOD=ONLINE_PHYSICAL` You can use one of these data transfer method values:

- `OSS` - Oracle Data Guard using Object Storage Service (OSS) for standby initialization.

  Supported for `PLATFORM_TYPE` set to Oracle Cloud Infrastructure (`VMDB`), Exadata Cloud Service (`EXACS`), and Exadata Cloud at Customer (`EXACC`).

  Also set `ZDM_LOG_OSS_PAR_URL` to the Cloud Object Store pre-authenticated URL if you want to upload migration logs onto Cloud Object Storage. For information about getting a pre-authenticated URL see Oracle Cloud documentation at https://docs.cloud.oracle.com/en-us/iaas/Content/Object/Tasks/usingpreauthenticatedrequests.htm#usingconsole.
- `EXTBACKUP` - Oracle Data Guard with existing backup in external location .

Supported for `PLATFORM_TYPE` set to Exadata Cloud at Customer (`EXACC`)

Also, create a standby control file backup in the specified path and provide read permissions to the backup pieces for the target database user. For example,

```
RMAN> BACKUP CURRENT CONTROLFILE FOR STANDBY FORMAT 'BACKUP_PATH/
lower_case_dbname/standby_ctl_%U';
```

Where standby_ctl_%U is a system-generated unique file name.

- `ZDLRA` - Oracle Data Guard using ZDLRA for standby initialization.

  Supported for `PLATFORM_TYPE` set to Exadata Cloud at Customer (`EXACC`), and set the following parameters.

  - Set `SRC_ZDLRA_WALLET_LOC` for the wallet location, for example,

    ```
    SRC_ZDLRA_WALLET_LOC=/u02/app/oracle/product/12.1.0/dbhome_3/dbs/zdlra
    ```

  - Set `TGT_ZDLRA_WALLET_LOC` for the wallet location, for example, `TGT_ZDLRA_WALLET_LOC=`*`target_database_oracle_home`*`/dbs/zdlra`.
  - Set `ZDLRA_CRED_ALIAS` for the wallet credential alias, for example,

    ```
    ZDLRA_CRED_ALIAS=zdlra_scan:listener_port/zdlra9:dedicated
    ```

- `NFS` - Oracle Data Guard using backup location such as NFS.

  Supported for `PLATFORM_TYPE` set to Exadata Cloud at Customer (`EXACC`)

  Also set `BACKUP_PATH` to specify the actual NFS path which is made accessible from both the source and target database servers, for example, an NFS mount point. The NFS mount path should be same for both source and target database servers. This path does not need to be mounted on the Zero Downtime Migration service host. Note the following considerations:

  - The source database is backed up to the specified path and restored to Exadata Cloud at Customer using RMAN SQL*Net connectivity.
  - The path set in `BACKUP_PATH` should have ‘rwx’ permissions for the source database user, and at least read permissions for the target database user.
  - In the path specified by `BACKUP_PATH`, the Zero Downtime Migration backup procedure will create a directory, `$BACKUP_PATH/`*`dbname`*, and place the backup pieces in this directory.

When `MIGRATION_METHOD=OFFLINE_PHYSICAL` You can use one of these data transfer method values:

- `OSS` - Migration using backup and restore through Object Storage Service, supported for Oracle Cloud Infrastructure (`VMDB`), Exadata Cloud Service (`EXACS`), and Exadata Cloud at Customer (`EXACC`). SQL*Net connectivity between source and target not needed.
- `NFS` - Migration using backup and restore through NFS, supported for Exadata Cloud at Customer (`EXACC`). SQL*Net connectivity between source and target not needed.

Additional Oracle Cloud Object Storage Settings

When `DATA_TRANSFER_MEDIUM=OSS`, set the following parameters to access the Oracle Cloud Object Storage.

The source database is backed up to the specified container and restored to Exadata Cloud at Customer using RMAN SQL*Net connectivity.

- Set `HOST` to the cloud storage REST endpoint URL.
  - For Oracle Cloud Infrastructure storage the typical value format is `HOST=https://swiftobjectstorage.us-phoenix-1.oraclecloud.com/v1/`*`ObjectStorageNamespace`*

    To find the Object Storage Namespace value, log in to the Cloud Console and select **Menu**, **Administration**, **Tenancy Detail**, and in the **Object Storage Settings** section find **Value against entry Object Storage Namespace:**
  - For Oracle Cloud Infrastructure Classic storage the typical value format is `HOST=https://acme.storage.oraclecloud.com/v1/Storage-`*`tenancy name`*
- Set the Object Storage bucket `OPC_CONTAINER` parameter.

  The bucket is also referred to as a container for Oracle Cloud Infrastructure Classic storage. Make sure that the Object Storage bucket is created using the Oracle Cloud Service Console as appropriate. Make sure adequate storage is provisioned and available on the object store to accommodate the source database backup.

TGT_SSH_TUNNEL_PORT

If SSH tunneling is set up, set the `TGT_SSH_TUNNEL_PORT` parameter.

Data and Redo Locations

Zero Downtime Migration automatically discovers the location for `data`, `redo`, and `reco` storage volumes from the specified target database. If you need to override the discovered values, specify the target database data files storage (ASM or ACFS) location using the appropriate set of parameters.

- ASM: `TGT_DATADG`, `TGT_REDODG`, and `TGT_RECODG`
- ACFS: `TGT_DATAACFS`, `TGT_REDOACFS`, and `TGT_RECOACFS`

SKIP_FALLBACK

Set `SKIP_FALLBACK=TRUE` if you do not want to ship redo logs from the target to the source standby, either voluntarily or because there is no connectivity between the target and the source.

TGT_SKIP_DATAPATCH

Zero Downtime Migration runs the datapatch utility by default as part of the migration process if the target database environment is at a higher patch level than the source database (for example, if the source database is at Jan 2020 PSU/BP and the target database is at April 2020 PSU/BP).

If you want to skip this task set the `TGT_SKIP_DATAPATCH=FALSE` response file parameter.

*PHASE_NAME*_MONITORING_INTERVAL

Set `PHASE_NAME_MONITORING_INTERVAL=n mins` if you want Zero Downtime Migration to monitor and report the status of backup and restore operations at the configured time interval during the migration. The default interval value is 10 minutes. To disable monitoring, set these values to 0 (zero).

```
ZDM_BACKUP_FULL_SRC_MONITORING_INTERVAL=
ZDM_BACKUP_INCREMENTAL_SRC_MONITORING_INTERVAL=
ZDM_BACKUP_DIFFERENTIAL_SRC_MONITORING_INTERVAL=
ZDM_CLONE_TGT_MONITORING_INTERVAL=
ZDM_OSS_RESTORE_TGT_MONITORING_INTERVAL=
ZDM_OSS_RECOVER_TGT_MONITORING_INTERVAL=
```

ZDM_BACKUP_RETENTION_WINDOW

Set `ZDM_BACKUP_RETENTION_WINDOW=number of days` if you wish to retain source database backup after the migration.

ZDM_SRC_TNS_ADMIN

Set `ZDM_SRC_TNS_ADMIN=TNS_ADMIN value` in case of custom location.

Migrating an On-Premises Database to an On-Premises Exadata Database Machine

An on-premises migration to an on-premises Exadata Database Machine target using Zero Downtime Migration works the same way as a migration to a cloud target. In the response file, you indicate that the migration target is on-premises by setting `PLATFORM_TYPE=NON_CLOUD`.

Just like in cloud migration scenarios, you must provision the target database with the shape and size desired, including configuring any initialization parameters, before starting the migration. The target database is expected to be the same major version as the source database, Oracle Grid Infrastructure is mandatory at the target database, and target datafiles can be stored on ASM or ACFS.

One aspect where an on-premises to on-premises migration is different from migrating to the cloud is in the handling of Transparent Data Encryption (TDE). On the cloud, TDE is mandatory for Oracle Database 12.2 and later releases; however, for an on-premises to on-premises migration, TDE must be configured at the target only if TDE is used at the source. You must configure TDE at the target before the migration starts; Zero Downtime Migration does not configure it for you.

You can specify that TDE is not configured at the source or target by setting the response file parameter `ZDM_TDE_MANDATORY=FALSE`. This parameter can only be used when you set `PLATFORM_TYPE=NON_CLOUD`. With `ZDM_TDE_MANDATORY=FALSE` set, Zero Downtime Migration does not require TDE at the target when the source is not using TDE, and does not encrypt the target on restore.

For an on-premises Exadata target database migration, `MIGRATION_METHOD` can be set to `ONLINE_PHYSICAL` or `OFFLINE_PHYSICAL`, and `DATA_TRANSFER_MEDIUM` can be set to any of the values supported by Zero Downtime Migration. Set the remaining parameters as you would for a cloud migration.

Converting a Non-CDB Database to a CDB During Migration

As part of the physical migration process, Zero Downtime Migration can handle conversion of a non-CDB source database to a PDB of the same version in the cloud. The conversion process transforms the source non-CDB into a target PDB that is then plugged into an existing CDB in the target.

Downtime will increase due to the non-CDB to CDB conversion. This process is offline (no redo transport and apply), and no rollback is possible.

Source non-CDB Database Prerequisites

- Oracle Database 12c or later versions, because this is when multitenant architecture became available
- Same character set as the target CDB

Target Database CDB and PDB Prerequisites

- The target CDB must not contain a PDB with same name as the resulting converted PDB, because Zero Downtime Migration will create the PDB.
- The target database must be at least the same major version as the source database.
  - If the minor version is different on the target, it must be a higher minor version than the source database.
  - If the patch level is different, you must set the response file parameter `TGT_SKIP_DATAPATCH=FALSE`.

Transparent Data Encryption Requirements

- Transparent Data Encryption (TDE) is optional on the source database. If TDE is not set there is no further information required; however if TDE is set up on source, the credentials for export and import TDE keys are required.
- For source credentials, the `migrate database` command must include either `-tdekeystorepasswd` or the `-tdekeystorewallet` option.
- If any of these options is used then the target credentials must be also provided by using either `-tgttdekeystorepasswd` or the `-tgttdekeystorewallet` option

Application Express Requirements

- If Application Express (APEX) is not installed on the source there are no further requirements.
- If APEX exists on the source, and the source database is a non-CDB, you must choose one of the following options:
  - Remove APEX from the source non-CDB.
  - Verify that the APEX version on the target CDB is the same as that on the source.
    If APEX is not at the same version conversion is not possible; APEX schemas vary between versions and the target PDB will not be able to open.

The target CDB is not dropped in the process, and the presence or absence of other PDBs does not affect the outcome of the conversion and plug-in.

Parameters in the response file are set as follows:

- (Required) NONCDBTOPDB_CONVERSION: Set to TRUE to indicate that you want to convert a source database from non-CDB to PDB.
- (Optional) NONCDBTOPDB_SWITCHOVER: Set to TRUE for a physical migration using Data Guard switchover, to execute switchover operations during a migration job with non-CDB to PDB conversion enabled.

The following are examples of the ZDMCLI migrate database command usage for converting a non-CDB to a PDB during migration using the TDE credentials:

```
zdmuser> $ZDM_HOME/bin/zdmcli migrate database
          -sourcedb source_db_unique_name_value -sourcenode
source_database_server_name -srcroot
          -targetnode target_database_server_name -backupuser
Object_store_login_user_name
          -rsp response_file_location -tgtauth zdmauth -tgtarg1
user:target_database_server_login_user_name
          -tgtarg2
identity_file:ZDM_installed_user_private_key_file_location -tgtarg3
sudo_location:/user/bin/sudo
          -tdekeystorepasswd -tgttdekeystorepasswd
```

```
zdmuser> $ZDM_HOME/bin/zdmcli migrate database
          -sourcedb source_db_unique_name_value -sourcenode
source_database_server_name -srcroot
          -targetnode target_database_server_name -backupuser
Object_store_login_user_name
          -rsp response_file_location -tgtauth zdmauth -tgtarg1
user:target_database_server_login_user_name
          -tgtarg2
identity_file:ZDM_installed_user_private_key_file_location -tgtarg3
sudo_location:/user/bin/sudo
          -tdekeystorepasswd -tgttdekeystorewallet /scratch/credentials/
cdbtde.sso
```

Preparing for a Logical Migration

The following topics describe how to configure the Zero Downtime Migration prerequisites before running a logical migration job.

Logical Migration Prerequisites

Complete the following prerequisites to prepare for a logical migration.

- Create an OCI API key pair. See Required Keys and OCIDs for details.
- Create an Object Store bucket on Oracle Cloud Infrastructure if you are using Object Storage as a data transfer medium. This is not required for Exadata Cloud at Customer or on-premises Exadata Database Machine targets.

- If the source database listener is configured with TLS (TCPS) using self-signed database server certificates, then ensure that the self-signed certificate is added to the Zero Downtime Migration home cert store as follows.

  ```
  keytool -import -keystore ZDM_HOME/jdk/jre/lib/security/cacerts -
  trustcacerts
  -alias "src ca cert" -file source_db_server-certificate
  ```

- If you are not using database link, ensure that the file system used for the Data Pump export directory has sufficient space to store Data Pump dump files.
- If you are using an existing database link between the target database to an on-premises source database by `global_name` of the source database, ensure that the DBLINK is not broken. Zero Downtime Migration can reuse the pre-existing DBLINK for migration if that data transfer medium is configured.
- For online migration, do the following:
  - Set up an Oracle GoldenGate Microservices hub.
    - For Oracle Cloud Service targets, deploy Oracle GoldenGate Microservices from Oracle Cloud Marketplace. See Deploying Oracle GoldenGate Microservices on Oracle Cloud Marketplace for details.
    - If you already own Oracle GoldenGate Microservices, you can create a deployment for the source and target using your pre-existing Oracle GoldenGate Microservices instance.
  - If the target database is configured to use SSL/TLS, then ensure that the wallet containing certificates for TLS authentication is located in the correct location on the GoldenGate instance, as follows:
    - For an Autonomous Database, the wallet file should be located in directory `/u02/deployments/`*`deployment_name`*`/etc/adb`
    - For a co-managed database, the wallet file should be located in directory `/u02/deployments/`*`deployment_name`*`/etc`

    Autonomous databases are always configured to use TLS.
  - If the source database is configured to use SSL/TLS, then ensure that the wallet containing certificates for TLS authentication is located in directory `/u02/deployments/deployment_name/etc` on the GoldenGate instance.
  - Complete the source database prerequisites. See Prepare the Source Database for Logical Migration
  - On the target database:

    If the target is Autonomous Database, unlock the pre-created `ggadmin` user.

    If the target is not Autonomous database, create a `ggadmin` user in the target PDB. See Prepare the Source Database for Logical Migration for information about creating this user.
- Ensure that the OCI network security rules allow the following connections:

Table 3-1 Prerequisite Connections for Online Logical Migration

Connection	Source	Destination
SQL*Net	GoldenGate hub	Source database
SQL*Net	GoldenGate hub	Target database
SQL*Net	ZDM server	Source database
SSH	ZDM server	Source database server
SQL*Net	ZDM server	Target database
HTTPS	ZDM server[1]	GoldenGate hub

[1]The Zero Downtime Migration server should be allowed to make HTTPS over port 443 calls to an OCI REST endpoint.

See Zero Downtime Migration Port Requirements for more information.

Prepare the Source Database for Logical Migration

Complete the following prerequisites on the source database to prepare for an online logical migration.

Offline and Online Migrations Require:

- The character set on the source database must be the same as the target database.
- Configure the streams pool with the initialization parameter `STREAMS_POOL_SIZE`.

  For offline logical migrations, for optimal Data Pump performance, it is recommended that you set `STREAMS_POOL_SIZE` to a minimum of 256MB-350MB, to have an initial pool allocated, otherwise you might see a significant delay during start up.

  For online logical migrations, set `STREAMS_POOL_SIZE` to at least 2GB. See https://support.oracle.com/epmos/faces/DocumentDisplay?id=2078459.1 for the recommendation 1GB `STREAMS_POOL_SIZE` per integrated extract + additional 25 percent.
- System time of the Zero Downtime Migration service host and source database server should be in sync with your Oracle Cloud Infrastructure target.

  If the time on any of these systems varies beyond 6 minutes from the time on OCI, it should be adjusted. You can use `ntp time check` to synchronize the time if NTP is configured. If NTP is not configured, then it is recommended that you configure it. If configuring NTP is not an option, then you need to correct the time manually to ensure it is in sync with OCI time.
- If you are using a database link, and your target database is on Autonomous Database Shared Infrastructure, you must configure TCPS on the source. Autonomous Database Shared Infrastructure doesn't allow a database link to a source that is not configured with TCPS.

Online Migrations Require:

- If the source is Oracle Database 11.2, apply mandatory 11.2.0.4 RDBMS patches on the source database.

  See My Oracle Support note Oracle GoldenGate -- Oracle RDBMS Server Recommended Patches (Doc ID 1557031.1)

  – Database PSU 11.2.0.4.200414 includes a fix for Oracle GoldenGate performance bug 28849751 - IE PERFORMANCE DEGRADES WHEN NETWORK LATENCY BETWEEN EXTRACT AND CAPTURE IS MORE THAN 8MS

  - OGG RDBMS patch 31704157 MERGE REQUEST ON TOP OF DATABASE PSU 11.2.0.4.200414 FOR BUGS 31182000 20448066 - This patch combines mandatory fixes for Oracle GoldenGate Microservices bug 20448066 DBMS_XSTREAM_GG APIS SHOULD BE ALLOWED FOR SCA PROCESSES and required OGG RDBMS patch 31182000 MERGE REQUEST ON TOP OF DATABASE PSU 11.2.0.4.200414 FOR BUGS 2990912 12668795.

    Although MOS note 1557031.1 mentions OGG patch 31177512, it conflicts with a patch for bug 20448066. As such, OGG patch 31704157 should be used instead of OGG patch 31177512.

- If the source is Oracle Database 12.1.0.2 or a later release, apply mandatory RDBMS patches on the source database.
  See My Oracle Support note Latest GoldenGate/Database (OGG/RDBMS) Patch recommendations (Doc ID 2193391.1), which lists the additional RDBMS patches needed on top of the latest DBBP/RU for Oracle Database 12c and later.
- Enable `ARCHIVELOG` mode for the database. See Changing the Database Archiving Mode.
- Enable `FORCE LOGGING` to ensure that all changes are found in the redo by the Oracle GoldenGate Extract process. See Specifying FORCE LOGGING Mode
- Enable database minimal supplemental logging. See Minimal Supplemental Logging.

```
SQL> ALTER DATABASE ADD SUPPLEMENTAL LOG DATA;
```

- Enable initialization parameter `ENABLE_GOLDENGATE_REPLICATION`.
- Install the `UTL_SPADV` or `UTL_RPADV` package for Integrated Extract performance analysis.

  See Collecting XStream Statistics Using the UTL_RPADV Package. Note that the package changes name from `UTL_SPADV` to `UTL_RPADV` in Oracle Database 19c.
- Create a GoldenGate administration user, `ggadmin`, granting all of the permissions listed in the example. If the source database is multitenant (CDB), create the user in the source PDB.

```
SQL> create user ggadmin identified by password default tablespace
users temporary tablespace temp;
SQL> grant connect, resource to ggadmin;
SQL> alter user ggadmin quota 100M ON USERS;
SQL> grant unlimited tablespace to ggadmin;
SQL> grant select any dictionary to ggadmin;
SQL> grant create view to ggadmin;
SQL> grant execute on dbms_lock to ggadmin;
SQL> exec dbms_goldengate_auth.GRANT_ADMIN_PRIVILEGE('ggadmin');
```

- If the source database is multitenant (CDB), also create user `c##ggadmin` in `CDB$ROOT` as shown here.

```
SQL> create user c##ggadmin identified by password default
tablespace users temporary tablespace temp;
SQL> grant connect, resource to c##ggadmin;
SQL> grant unlimited tablespace to c##ggadmin;
SQL> alter user c##ggadmin quota 100M ON USERS;
SQL> grant select any dictionary to c##ggadmin;
SQL> grant create view to c##ggadmin;
SQL> grant execute on dbms_lock to c##ggadmin;
SQL> exec
dbms_goldengate_auth.GRANT_ADMIN_PRIVILEGE('c##ggadmin',container=>'all');
```

- During the migration period, to provide the most optimal environment for fast database replication, avoid large batch DML operations. Running large batch operations, like a single transaction that affects multi-millions of rows, can slow down replication rates. Create, alter, and drop DDL operations are not replicated.

Offline Migrations Require:

- The `DATAPUMP_EXP_FULL_DATABASE` and `DATAPUMP_IMP_FULL_DATABASE` roles are required. These roles are required for Data Pump to determine whether privileged application roles should be assigned to the processes comprising the migration job.

  `DATAPUMP_EXP_FULL_DATABASE` is required for the export operation at the source database for the specified database user. The `DATAPUMP_IMP_FULL_DATABASE` role is required for the import operation at the specified target database for specified target database user.

  See the Oracle Data Pump documentation for more information.

Preparing the Logical Migration Response File

Set the required logical migration response file parameters. Get the response file template, `$ZDM_HOME/rhp/zdm/template/zdm_logical_template.rsp`, which is used to create your Zero Downtime Migration response file for the database migration procedure, and edit the file as described here.

The logical migration response file settings are described in detail in Zero Downtime Migration Logical Migration Response File Parameters Reference.

The following parameters are required for an offline or online logical migration:

- `MIGRATION_METHOD`: Set to `ONLINE_LOGICAL` for online migration with GoldenGate or `OFFLINE_LOGICAL` for an offline Data Pump transfer.
- `DATA_TRANSFER_MEDIUM`: Set to `OSS` for Object Storage bucket, `NFS` for a shared Network File System, `DBLINK` for a direct transfer using a database link, or `COPY` to use secure copy.

  Unless you are using the default data transfer servers for handling the Data Pump dumps, you may also need to configure the data transfer node settings for the source and target database environments.

  See Configuring the Transfer Medium and Specifying Transfer Nodes for details.
- For an offline logical migration of an Oracle Database 11g source to an 11g target, set `DATAPUMPSETTINGS_SECUREFILELOB=FALSE` or you may get errors.
- Set the following target database parameters.
  - `TARGETDATABASE_OCID` specifies the Oracle Cloud resource identifier.

    For example:
    ocid1.instance.oc1.phx.abuw4ljrlsfiqw6vzzxb43vyypt4pkodawglp3wqxjqofakrwvou52gb6s5a

See also https://docs.cloud.oracle.com/en-us/iaas/Content/General/Concepts/identifiers.htm

- `TARGETDATABASE_ADMINUSERNAME` specifies the database administrator user name. For example, for a co-managed database migration user name as `system` and for an Autonomous Database migration user name as `admin`.

- Set the following source database parameters.
  - `SOURCEDATABASE_ADMINUSERNAME` specifies the database administrator user name. For example, user name as `system`.
  - `SOURCEDATABASE_CONNECTIONDETAILS_HOST` specifies the listener host name or IP address. In case of Oracle RAC, the SCAN name can be specified. (not required for Autonomous Database)
  - `SOURCEDATABASE_CONNECTIONDETAILS_PORT` specifies the listener port number. (not required for Autonomous Database)
  - `SOURCEDATABASE_CONNECTIONDETAILS_SERVICENAME` specifies the fully qualified service name. (not required for Autonomous Database)

    For example: *service_name.DB_domain*

    See also https://docs.cloud.oracle.com/en-us/iaas/Content/Database/Tasks/connectingDB.htm
- Set the following `OCIAUTHENTICATIONDETAILS` parameters.

  For more information about the required settings, see https://docs.cloud.oracle.com/en-us/iaas/Content/API/Concepts/apisigningkey.htm#RequiredKeysandOCIDs
  - `OCIAUTHENTICATIONDETAILS_USERPRINCIPAL_TENANTID` specifies the OCID of the OCI tenancy. You can find this value in the Console under Governance and Administration, Administration, Tenancy Details. The tenancy OCID is shown under Tenancy Information.

    For example:
    ocid1.tenancy.oc1..aaaaaaaaba3pv6wkcr4jqae5f44n2b2m2yt2j6rx32uzr4h25vqstifsfdsq

    See also https://docs.cloud.oracle.com/en-us/iaas/Content/Identity/Tasks/managingtenancy.htm
  - `OCIAUTHENTICATIONDETAILS_USERPRINCIPAL_USERID` specifies the OCID of the IAM user. You can find this value in the Console under Profile, User Settings.

    See also https://docs.oracle.com/en-us/iaas/Content/Identity/Tasks/managingusers.htm
  - `OCIAUTHENTICATIONDETAILS_USERPRINCIPAL_FINGERPRINT` specifies the fingerprint of the public API key.
  - `OCIAUTHENTICATIONDETAILS_USERPRINCIPAL_PRIVATEKEYFILE` specifies the absolute path of API private key file.
  - `OCIAUTHENTICATIONDETAILS_REGIONID` specifies the OCI region identifier.

    See the Region Identifier column in the table at https://docs.cloud.oracle.com/en-us/iaas/Content/General/Concepts/regions.htm

Oracle GoldenGate Settings

For online logical migrations, in addition to the above, you must also set the GoldenGate parameters, `TARGETDATABASE_GGADMINUSERNAME`, `SOURCEDATABASE_GGADMINUSERNAME`, `SOURCECONTAINERDATABASE_GGADMINUSERNAME`, and the parameters prefixed with `GOLDENGATEHUB` and `GOLDENGATESETTINGS`.

See Zero Downtime Migration Logical Migration Response File Parameters Reference for details about these parameters.

Oracle Data Pump Settings

Zero Downtime Migration automatically sets optimal defaults for Data Pump parameters to achieve better performance and ensure data security. If you need to further tune performance, there are several Data Pump settings that you can configure in the response file.

The default `DATAPUMPSETTINGS_JOBMODE=SCHEMA` is recommended for migrations to Autonomous Database.

See Oracle Data Pump Settings for Zero Downtime Migration for information about the default Data Pump property settings, how to select schemas or objects for inclusion or exclusion, and Data Pump error handling.

See Zero Downtime Migration Logical Migration Response File Parameters Reference for all of the Data Pump parameters you can set through Zero Downtime Migration.

Configuring the Transfer Medium and Specifying Transfer Nodes

Zero Downtime Migration offers various transfer options to make Oracle Data Pump dumps available to the target database server.

Using the `DATA_TRANSFER_MEDIUM` response file parameter you can configure the following data transfer methods:

- `OSS`: Oracle Cloud Object Storage.

  Supported for all migration types and targets.

- `NFS`: Network File System

  Supported for offline migrations to co-managed target database only.

- `DBLINK`: Direct data transfer from the source to the target over a database link.

  Supported for online and offline migrations to Autonomous Database Shared (Data Warehouse or Transaction Processing) and co-managed targets only.

- `COPY`: Transfer dumps to the target transfer node using secure copy.

  Supported for offline migrations to co-managed target databases only.

> **Note:**
>
> To take advantage of parallelism and achieve the best data transfer performance, Oracle recommends that you transfer data using `OSS` or `NFS` for databases over 50GB in size. The `DBLINK` transfer medium can be convenient for smaller databases, but this choice may involve uncertainty in performance because of its dependence on network bandwidth for the duration of the transfer.

Once the export of dumps on the source is completed, the dumps are uploaded or transferred in parallel as defined by parameter `DUMPTRANSFERDETAILS_PARALLELCOUNT` (defaults to 3), and any transfer failures are retried by default as specified in the parameter `DUMPTRANSFERDETAILS_RETRYCOUNT` (defaults to 3).

The transfer of dumps can be done from any node at the source data center, provided that the dumps are accessible from the given node. It is crucial to ascertain the network connectivity and transfer workload impact on the source database server in order to decide which data transfer approach to take.

Direct Transfer from Source to Target

This option applies only to on-premises Exadata Database Machine and co-managed cloud target databases.

Zero Downtime Migration enables logical migration using direct transfer of the Data Pump dump from the source to the target securely. The data is copied over from the source database directory object path to the target database server directory object path, or to a target transfer node, using either secure copy or RSYNC. This avoids the data being transferred over a WAN or needing additional shared storage between the source and target environments. This capability greatly simplifies the logical migration within the data center.

About Transfer Nodes

You will configure a node, referred as a **transfer node**, for both the source data center and the target tenancy.

The response file parameters that are prefixed with `DUMPTRANSFERDETAILS_SOURCE_TRANSFERNODE` designate the node that handles the export dumps at the source data center. This **source transfer node** defaults to the source database.

Similarly, the response file parameters that are prefixed with `DUMPTRANSFERDETAILS_TARGET_TRANSFERNODE` designate the node that handles the import of dumps at the target. This **target transfer node** defaults to the target database, for co-managed targets.

Transfer Node Requirements

The source transfer node can be any of the following:

- Source database server (default)
- NAS mounted server
- Zero Downtime Migration service node

The target transfer node can be any of the following:

- Target Database server (default)
- NAS mounted server
- Zero Downtime Migration service node

For a server to be designated as transfer node, the following critical considerations are necessary.

- Availability of CPU and memory to process the upload or transfer workload
- Connectivity to the specified upload or transfer target

    - Port 443 connectivity to Object Storage Service if the chosen data transfer medium is `OSS`
    - Port 22 connectivity to target storage server if the chosen transfer medium is `COPY`
- Availability of Oracle Cloud Infrastructure CLI. For speedier and resilient upload of dumps this is the recommended transfer utility for the `OSS` transfer medium.
- OCI CLI must be installed and configured as detailed in https://docs.oracle.com/en-us/iaas/Content/API/SDKDocs/cliinstall.htm.

  Installing and configuring OCI CLI on each source database server may not be feasible. In such cases, one of the nodes in the data center can be designated as a transfer node with OCI CLI configured, and this node can share a network storage path with the database servers for Data Pump dumps to be created. This also avoids the upload workload consuming additional CPU and memory on production database servers.

The designated transfer node can act as the gateway server at the data center for the external data transfer allowing transfer data traffic, thus avoiding the need to allow data transfer from the source database server or to the target database server.

Optionally, the additional transfer node requirement can be avoided by leveraging the Zero Downtime Migration server as the transfer node, provided that the Zero Downtime Migration service is placed in an on-premises data center and can meet the transfer node requirements listed above.

Using the Oracle Cloud Object Storage Transfer Medium

Object Storage data transfer medium is supported for all migration types and targets.

When using Object Storage as the data transfer medium, by setting `DATA_TRANSFER_MEDIUM=OSS`, it is recommended that dumps be uploaded using OCI CLI for faster and more secure and resilient uploads. You must configure OCI CLI in the upload node, and set parameter `DUMPTRANSFERDETAILS_SOURCE_USEOCICLI` to `TRUE`, the parameters for OCI CLI are

`DUMPTRANSFERDETAILS_SOURCE_USEOCICLI`

`DUMPTRANSFERDETAILS_SOURCE_OCIHOME`

Using the Database Link Transfer Medium

Supported for online and offline migrations to Autonomous Database Shared (Data Warehouse or Transaction Processing) and co-managed targets only.

When you set `DATA_TRANSFER_MEDIUM=DBLINK`, a database link is created from the OCI co-managed database or Autonomous Database target to the source database using the `global_name` of the specified source database.

Zero Downtime Migration creates the database link if it does not already exist, and the link is cleaned once the Data Pump import phase is complete.

Using the NFS Transfer Medium

Supported for offline migrations to co-managed target database only.

The NFS mode of transfer is available, by setting `DATA_TRANSFER_MEDIUM=NFS`, for co-managed target databases that avoid the transfer of dumps. You should ensure that the specified path is accessible between the source and target database server path.

Zero Downtime Migration ensures the security of dumps in the shared storage by preserving the restricted permission on the dumps such that only the source and target database users are allowed to access the dump.

Using the Copy Transfer Medium

Supported for offline migrations to co-managed target databases only.

Dumps can be transferred from the source to the target securely, by setting `DATA_TRANSFER_MEDIUM=COPY`. The relevant parameters are as follows:

`DUMPTRANSFERDETAILS_TRANSFERTARGET_USER`

`DUMPTRANSFERDETAILS_TRANSFERTARGET_USERKEY`

`DUMPTRANSFERDETAILS_TRANSFERTARGET_HOST`

`DUMPTRANSFERDETAILS_TRANSFERTARGET_SUDOPATH`

`DUMPTRANSFERDETAILS_TRANSFERTARGET_DUMPDIRPATH`

You can leverage the RSYNC utility instead of SCP. Set `DUMPTRANSFERDETAILS_RSYNCAVAILABLE` to `TRUE`, and verify that RSYNC is available both at the source and target transfer nodes.

Configuring Resiliency to Intermittent Network Failures

Zero Downtime Migration is resilient to intermittent network failures that can cause backups or SSH connectivity to fail.

Physical Migration Resiliency

Zero Downtime Migration can auto-detect intermittent network failures. Zero Downtime Migration automatically retries the RMAN retry-able errors, and some retry customization is available.

SSH connection retries are customized using the following parameters:

`SRC_SSH_RETRY_TIMEOUT`

`TGT_SSH_RETRY_TIMEOUT`

You can customize RMAN backup retries with following parameters:

`ZDM_OPC_RETRY_WAIT_TIME`

`ZDM_OPC_RETRY_COUNT`

`ZDM_OPC_RETRY_WAIT_TIME`

Logical Migration Resiliency

Intermittent network failures observed during the transfer of Data Pump dumps can mitigated by setting the `DUMPTRANSFERDETAILS_RETRYCOUNT` parameter. The default value is 3.

Database Server Connectivity Using a Bastion Host

Zero Downtime Migration lets you configure connectivity to the source and target database servers through a bastion host for both physical and logical migration work flows.

Note that a bastion host cannot be used to connect to an Autonomous Database, except for JDBC connections.

Use the following sections to configure the appropriate parameters for physical and logical migrations that must connect to the source or target database server through a bastion host.

SSH Connection to Database Servers

Connecting database servers through a bastion host requires the following information:

- **Bastion Host IP Address and Source Database Server IP Address:** To connect to the database server through a bastion host, the bastion host IP address and the source node IP address are required.
- **Bastion Port Number:** The port number defaults to 22 if not specified.
- **Bastion User:** The bastion host user is only required if the user specified for the argument `zdmauth` plug-in is different from the user of the bastion host. The bastion user defaults to the user specified for the source `zdmauth` plug-in if the property is not specified.
- **Bastion Identity File:** If the *`SRC/TGT`*`_BASTION_IDENTITY_FILE` parameter is not specified, the value defaults to the value specified for the `identity_file` argument of the `zdmauth` plug-in argument.

Physical Migration Response File Parameters

Configure the following response file parameters for a physical migration.

Source Database Server

`SRC_BASTION_HOST_IP=`

`SRC_BASTION_PORT=`

`SRC_BASTION_USER=`

`SRC_BASTION_IDENTITY_FILE=`

`SRC_HOST_IP=`

Target Database Server

`TGT_BASTION_HOST_IP=`

`TGT_BASTION_PORT=`

`TGT_BASTION_USER=`

`TGT_BASTION_IDENTITY_FILE=`

`TGT_HOST_IP=`

Logical Migration Response File Parameters

Configure the following response file parameters for a logical migration.

Source Database Server

`SOURCECONTAINERDATABASE_CONNECTIONDETAILS_BASTIONDETAILS_IP=`

`SOURCECONTAINERDATABASE_CONNECTIONDETAILS_BASTIONDETAILS_PORT=22`

`SOURCECONTAINERDATABASE_CONNECTIONDETAILS_BASTIONDETAILS_IDENTITYFILE=`

`SOURCECONTAINERDATABASE_CONNECTIONDETAILS_BASTIONDETAILS_USERNAME=`

`SOURCECONTAINERDATABASE_CONNECTIONDETAILS_BASTIONDETAILS_REMOTEHOSTIP=`

Target Database Server (including Autonomous Database)

`TARGETDATABASE_CONNECTIONDETAILS_BASTIONDETAILS_IP=`

`TARGETDATABASE_CONNECTIONDETAILS_BASTIONDETAILS_PORT=22`

`TARGETDATABASE_CONNECTIONDETAILS_BASTIONDETAILS_IDENTITYFILE=`

`TARGETDATABASE_CONNECTIONDETAILS_BASTIONDETAILS_USERNAME=`

`TARGETDATABASE_CONNECTIONDETAILS_BASTIONDETAILS_REMOTEHOSTIP=`

Preparing for Automatic Application Switchover

To minimize or eliminate service interruptions on the application after you complete the database migration and switchover, prepare your application to automatically switch over connections from the source database to the target database.

Note:

In physical migrations, Autonomous Database targets are not supported for automatic application switchover.

In the following example connect string, the application connects to the source database, and when it is not available the connection is switched over to the target database.

```
(DESCRIPTION=
    (FAILOVER=on)(LOAD_BALANCE=on)(CONNECT_TIMEOUT=3)(RETRY_COUNT=3)
    (ADDRESS_LIST=
        (ADDRESS=(PROTOCOL=TCP)(HOST=source_database_scan)(PORT=1521))
        (ADDRESS=(PROTOCOL=TCP)(HOST=target_database_scan)(PORT=1521)))
    (CONNECT_DATA=(SERVICE_NAME=zdm_prod_svc)))
```

On the source database, create the service, named zdm_prod_svc in the examples.

```
srvctl add service -db clever -service zdm_prod_svc -role PRIMARY
 -notification TRUE -session_state dynamic -failovertype transaction
 -failovermethod basic -commit_outcome TRUE -failoverretry 30 -
failoverdelay 10
 -replay_init_time 900 -clbgoal SHORT -rlbgoal SERVICE_TIME -preferred
clever1,clever2
 -retention 3600 -verbose
```

If the db_domain changes between the source and target then the connect string specified in the application should cater to both for failover to be effective.

```
(DESCRIPTION_LIST=
               (FAILOVER=ON)
               (LOAD_BALANCE=ON)
               (DESCRIPTION=
               (ADDRESS= (PROTOCOL=TCP) (HOST=SRC_SCAN) (PORT=1521))
               (CONNECT_DATA=
               (SERVICE_NAME=SVC.SRC_DOMAIN)))
               (DESCRIPTION=
               (ADDRESS= (PROTOCOL=TCP) (HOST=TGT_SCAN) (PORT=1521))
               (CONNECT_DATA=
               (SERVICE_NAME= SVC.TGT_DOMAIN))
```

See Also:

Oracle MAA white papers about client failover best practices on the Oracle Active Data Guard Best Practices page at https://www.oracle.com/goto/maa
High Availability in *Oracle Database Development Guide*

Customizing a Migration Job

You can customize the Zero Downtime Migration work flow with scripts which can be run at the beginning or end of a specified migration job phase. In Zero Downtime Migration, these customizations are called **custom plug-ins with user actions**.

The following topics describe how to customize a migration job.

About Custom Plug-ins with User Actions

In Zero Downtime Migration, a custom script, or bundle of scripts, that you want to plug in and run as part of a migration job is called a **custom plug-in with user action**.

A custom plug-in with user action, which is also referred to as a **user action**, is associated with an operational phase in the migration job and can be run before the phase or after it.

A **pre-action** is a user action performed at the beginning of the associated phase. Likewise, a **post-action** is performed at the end of the associated phase.

Once a user action is associated with a migration phase, Zero Downtime Migration copies it to the respective node (for non-Autonomous Databases) and run the user action locally. Zero Downtime Migration supplies the user action script execution with a set of parameters for developers to use the DBNAME, DBHOME, DB SCAN and ZDM log locations, and many more parameters.

For Autonomous Database, Zero Downtime Migration allows SQL to be registered as a user action and executes the SQL in an Autonomous Database instance from the Zero Downtime Migration server through a JDBC connection.

Parameters Supplied for Custom Plug-ins with User Actions

At run time, Zero Downtime Migration invokes the user action script and supplies it with set of auto-populated ENV variables that can be used to program the logic.

These variables are supplied as ENV variables with values so you can program the custom plug-in using these values.

For example, the value of `ZDM_SRCDBHOME` specifies the database home associated with the current migration job source database, and similarly `SRC_SCAN_NAME` indicates the scan listener information for the source database that you can use to connect to the database.

The following is a listing of the user action parameters with expected values. See the footnotes for more details.

```
RHP_OPTYPE=MIGRATE_DATABASE
RHP_PHASE=PRE
ZDM_SRCDB=src_db_name
ZDM_SRCDBHOME=src_db_home
ZDM_TARGETDB=tgt_db_name
ZDM_TARGETDBHOME=tgt_db_home
RHP_PROGRESSLISTENERHOST=zdm_node_name
RHP_PROGRESSLISTENERPORT=zdm_progress_listener_port
LOG_PATH=src/tgt_zdm_log_path1
SRC_SCAN_NAME=src_scan_name
SRC_SCAN_PORT=src_scan_port
TGT_SCAN_NAME=tgt_scan_name
TGT_SCAN_PORT=tgt_scan_port
RHP_USERACTIONDATA=user_action_data2
RHP_OP_PHASE=current_job_phase3
```

[1] `LOG_PATH` specifies the source or target node Zero Downtime Migration log path for the custom plug-in to store any log files for future reference.

[2] `RHP_USERACTIONDATA` specifies the useraction argument that was specified in `zdmcli migrate database` command option `-useractiondata` *`user_action_data`*

[3] `RHP_OP_PHASE` specifies the current phase of the migration job, for example,`ZDM_VALIDATE_SRC`.

User Action Scripts

You can use the following example user action scripts to help you create your own scripts.

For an on-premises source and Cloud target with node access, Zero Downtime Migration requires that the user action be a script. The script file is copied over to the respective node and is executed locally.

For Autonomous Database targets, Zero Downtime Migration requires that the user action script be a SQL file (.sql), which is executed in the Autonomous Database target using a JDBC connection.

Example 3-1 action.sh

Shown here is a sample user action script that discovers the supplied parameters.

```
    #!/bin/sh

    for var in $@
    do
    if [[ ${var} == *"ZDM_SRCDB="* ]]
    then
    IFS='=' read -ra DBARR <<< "${var}"
    SRCDBNAME=${DBARR[1]}
    fi
    if [[ ${var} == *"ZDM_TARGETDB="* ]]
    then
    IFS='=' read -ra DBARR <<< "${var}"
    TARGETDBNAME=${DBARR[1]}
    fi
    if [[ $var == *"ZDM_SRCDBHOME="* ]]
    then
    IFS='=' read -ra PATHARR <<< "$var"
    SRCDBHOME=${PATHARR[1]}
    fi
    if [[ $var == *"ZDM_TARGETDBHOME="* ]]
    then
    IFS='=' read -ra PATHARR <<< "$var"
    TARGETDBHOME=${PATHARR[1]}
    fi
    if [[ $var == *"RHP_OP_PHASE="* ]]
    then
    IFS='=' read -ra PHASEARR <<< "$var"
    ZDMPHASE=${PHASEARR[1]}
    fi
    if [[ $var == *"eval"* ]]
    then
    IFS='=' read -ra PATHARR <<< "$var"
    EVALRUN=${PATHARR[1]}
    fi
    if [[ $var == *"LOG_PATH"* ]]
    then
    IFS='=' read -ra PATHARR <<< "$var"
    LOGPATH=${PATHARR[1]}
    fi
    done

    echo "`date` Starting CUSTOM_USERACTION" >> $LOG_PATH/
CUSTOM_USERACTION.log
    echo $@ >> $LOG_PATH/CUSTOM_USERACTION.log
```

Example 3-2 Running SQL*Plus in a user action

```
#!/bin/sh

for var in $@
do
```

```
if [[ ${var} == *"ZDM_SRCDB="* ]]
then
IFS='=' read -ra DBARR <<< "${var}"
SRCDBNAME=${DBARR[1]}
fi
if [[ ${var} == *"ZDM_TARGETDB="* ]]
then
IFS='=' read -ra DBARR <<< "${var}"
TARGETDBNAME=${DBARR[1]}
fi
if [[ $var == *"ZDM_SRCDBHOME="* ]]
then
IFS='=' read -ra PATHARR <<< "$var"
SRCDBHOME=${PATHARR[1]}
fi
if [[ $var == *"ZDM_TARGETDBHOME="* ]]
then
IFS='=' read -ra PATHARR <<< "$var"
TARGETDBHOME=${PATHARR[1]}
fi
if [[ $var == *"RHP_OP_PHASE="* ]]
then
IFS='=' read -ra PHASEARR <<< "$var"
ZDMPHASE=${PHASEARR[1]}
fi
if [[ $var == *"eval"* ]]
then
IFS='=' read -ra PATHARR <<< "$var"
EVALRUN=${PATHARR[1]}
fi
if [[ $var == *"LOG_PATH"* ]]
then
IFS='=' read -ra PATHARR <<< "$var"
LOGPATH=${PATHARR[1]}
fi
done

    check_dv_staus()
    {
        export ORACLE_HOME=${2}
        export PATH=$ORACLE_HOME/bin:$PATH
        export LOGFILE=$3
        export currenthostname=`hostname -a` >> $LOGFILE
        echo "Current DB Host=$currenthostname" >> $LOGFILE
        CURRENTNODE=`srvctl status database -db ${1} |
grep $currenthostname | cut -d" " -f2` >> $LOGFILE
        SQLRETURN=$?
        echo "Curent DB Node=${CURRENTNODE}" >> $LOGFILE
        if [ "$SQLRETURN" -ne "0" ]; then
            return 1
        fi
        export ORACLE_SID=${CURRENTNODE}
        echo "`date` Checking Database Vault Status" >> $LOGFILE
        $ORACLE_HOME/bin/sqlplus -s / as sysdba 2>> $LOGFILE << EOF
> /dev/null
```

```
        whenever sqlerror exit 1
        SET PAGESIZE 60
        SET LINESIZE 1300
        SET VERIFY OFF TRIMSPOOL ON  HEADING OFF  TERMOUT OFF  FEEDBACK OFF
        spool ${LOGFILE} append;

        SELECT 'Check Status : '||PARAMETER FROM V\$OPTION WHERE PARAMETER =
'Oracle Database Vault' AND VALUE='TRUE';
        SELECT 'Check Grant : '||GRANTED_ROLE from dba_role_privs where
GRANTED_ROLE in ('DV_PATCH_ADMIN') and grantee='SYS';
        select distinct ('Check TDE enabled ' || ENCRYPTED) from
dba_tablespaces where ENCRYPTED='YES';
        spool off
        EOF
        SQLRETURN=$?
        if [ "$SQLRETURN" -ne "0" ]; then
            return 1
        fi

        if grep -q "Check Status : Oracle Database Vault" $LOGFILE;
        then
            echo "`date`:$ORACLE_SID:Database Vault is enabled " >> $LOGFILE
        if grep -q "Check Grant : DV_PATCH_ADMIN" $LOGFILE;
        then
            return 3 # sys privs already granted
        else
            return 4 # sys privs are not granted
        fi
        else
            echo "`date`:$ORACLE_SID: DV is not enabled" >> $LOGFILE
        return 2
        fi
    }

if [[ $ZDMPHASE == "ZDM_VALIDATE_SRC" || $ZDMPHASE == "ZDM_VALIDATE_TGT" ]]
then
  export LOGFILE=$LOG_PATH/${SRCDBNAME}_${ZDMPHASE}_datavault.log
  echo "`date` Start User Action for Phase: $ZDMPHASE " > $LOGFILE
  echo $@ >> $LOGFILE
  check_dv_staus $SRCDBNAME $SRCDBHOME $LOGFILE
  CHECKDV_STATUS_RETURN_CODE=$?
  if [ "$CHECKDV_STATUS_RETURN_CODE" -eq "1" ]; then
  echo "`date`:${SRCDBNAME}:Unable to check DataVault status" >> $LOGFILE
  exit 1
  fi

  if [ "$CHECKDV_STATUS_RETURN_CODE" -eq "2" ]; then
    echo "`date`:${SRCDBNAME}:DataVault is not enabled " >> $LOGFILE
    exit 0
  fi

  if [ "$CHECKDV_STATUS_RETURN_CODE" -eq "3" ];
  then
     echo "`date`:${SRCDBNAME}:Database Vault SYS privileges granted"
>> $LOGFILE
```

```
      exit 0
   fi
```

Example 3-3   Action file archive extractor action_extract.sh

If you bundle more than one user action into an action file, as described in Registering User Actions, you must also supply a script that unzips the action file, as shown in this example.

```
#!/bin/sh
MKDIR=/bin/mkdir
DATE=/bin/date
UNZIP=/usr/bin/unzip
#get the current location, extract path from it and then from the same
directory perform unzip to /tmp directory
script_path=$0
script_dir=${script_path%/*}
timestamp=$($DATE +%s)
unzip_dir=/tmp/rhp_${timestamp}
$MKDIR $unzip_dir
$UNZIP ${script_dir}/pack.zip -d $unzip_dir

echo "/bin/sh ${unzip_dir}/pack/main.sh $@"
/bin/sh ${unzip_dir}/pack/main.sh  "$@"
```

Once the archive is extracted, action_extract.sh runs main.sh, which in turn runs any user action scripts.

```
 #!/bin/sh
script_path=$0
script_dir=${script_path%/*}
#extract args and execute all scripts
echo "/bin/sh ${script_dir}/wc_add_pre.sh $@"
/bin/sh ${script_dir}/wc_add_pre.sh  "$@"
```

action_phase_pre.sh

```
#!/bin/sh
touch /tmp/SAMPLE_PRE_OUT.txt;
echo $* >> /tmp/SAMPLE_PRE_OUT.txt;
```

Registering User Actions

User actions must be registered with the Zero Downtime Migration service host to be plugged in as customizations for a particular operational phase.

The ZDMCLI `add useraction` command registers a custom action script that needs to be run in the source or target server. The user action script is copied to the Zero Downtime Migration metadata. You can use the `modify useraction` command if the script needs to be modified. An action script can be part of any number of custom actions in given migration job.

Determine the migration job phase the action has to be associated with, and run the ZDMCLI command ADD USERACTION, specifying -optype MIGRATE_DATABASE and the respective phase of the operation, whether the plug-in is run -pre or -post relative to that phase, and any on-error requirements.

You can register custom plug-ins for operational phases after ZDM_SETUP_TGT in the migration job work flow.

If the user action encounters an error at runtime, a behavior can be specified with the -onerror option, which you can set to either ABORT, to end the process, or CONTINUE, to continue the migration job even if the custom plug-in exits with an error.

Use the Zero Downtime Migration software installed user (for example, zmduser) to add user actions to a database migration job.

The following example shows you how to add user actions zdmvaltgt and zdmvalsrc.

```
zdmuser> $ZDM_HOME/bin/zdmcli add useraction -useraction zdmvaltgt -optype
MIGRATE_DATABASE
-phase ZDM_VALIDATE_TGT -pre -onerror ABORT -actionscript /home/zdmuser/
useract.sh

zdmuser> $ZDM_HOME/bin/zdmcli add useraction -useraction zdmvalsrc -optype
MIGRATE_DATABASE
-phase ZDM_VALIDATE_SRC -pre -onerror CONTINUE -actionscript /home/zdmuser/
useract1.sh
```

In the above command, the scripts useract.sh and useract1.sh, specified in the -actionscript option, are copied to the Zero Downtime Migration service host repository meta data, and they are run if they are associated with any migration job run using an action template.

Running Multiple Scripts in One Action

If you need to run multiple scripts as a single pre or post action, all of the action scripts can be bundled as an **action file** archive and supplied along with an action script, as shown in the following example. The action script should unzip the action file and invoke the scripts within.

```
zdmuser> $ZDM_HOME/bin/zdmcli add useraction -useraction
reconfigure_services
 -optype MIGRATE_DATABASE -phase ZDM_RECOVER_TGT -post -onerror ABORT
 -actionscript /home/zdmuser/action_extract.sh
 -actionfile /home/zdmuser/pack.zip
```

For example scripts and script requirements, see User Action Scripts.

Creating an Action Template

After the useraction plug-ins are registered, you create an action template that combines a set of action plug-ins which can be associated with a migration job.

An action template is created using the ZDMCLI command add imagetype, where the image type, imagetype, is a bundle of all of the useractions required for a specific type of database migration. Create an image type that associates all of the useraction plug-ins needed for the migration of the database. Once created, the image type can be reused for all migration operations for which the same set of plug-ins are needed.

The base type for the image type created here must be `CUSTOM_PLUGIN`, as shown in the example below.

For example, you can create an image type `ACTION_ZDM` that bundles both of the useractions created in the previous example, zdmvalsrc and zdmvaltgt.

```
zdmuser> $ZDM_HOME/bin/zdmcli add imagetype -imagetype ACTION_ZDM -
basetype
CUSTOM_PLUGIN -useractions zdmvalsrc,zdmvaltgt
```

Updating Action Plug-ins

You can update action plug-ins registered with the Zero Downtime Migration service host.

The following example shows you how to modify the useraction zdmvalsrc to be a `-post` action, instead of a `-pre` action.

```
zdmuser> $ZDM_HOME/bin/zdmcli modify useraction -useraction zdmvalsrc -
phase ZDM_VALIDATE_SRC
 -optype MIGRATE_DATABASE -post
```

This change is propagated to all of the associated action templates, so you do not need to update the action templates.

Querying Action Plug-ins

You can query action plug-ins registered with the Zero Downtime Migration service host.

To display the configuration of a user action:

```
zdmuser> $ZDM_HOME/bin/zdmcli query useraction -h
```

See query useraction for usage information.

Associating an Action Template with a Migration Job

When you run a migration job you can specify the image type that specifies the plug-ins to be run as part of your migration job.

As an example, run the migration command specifying the action template ACTION_ZDM created in previous examples, `-imagetype ACTION_ZDM`, including the image type results in running the useract.sh and useract1.sh scripts as part of the migration job workflow.

By default, the action plug-ins are run for the specified operational phase on all nodes of the cluster. If the access credential specified in the migration command option `–tgtarg2` is unique for a specified target node, then an additional auth argument should be included to specify the auth credentials required to access the other cluster nodes.

For example, specify `-tgtarg2 nataddrfile:`*`auth_file_with_node_and_identity_file_mapping`*.

A typical nataddrfile for a 2 node cluster with node1 and node2 is shown here.

```
node1:node1:identity_file_path_available_on_zdmservice_node
node2:node2:identity_file_path_available_on_zdmservice_node
```

4

Migrating Your Database with Zero Downtime Migration

Evaluate the database migration job, run the job, and perform other operations during and after a database migration.

See the Zero Downtime Migration Release Notes for the latest information about known issues, My Oracle Support notes, and runbooks.

Evaluate the Migration Job

Zero Downtime Migration provides options and tools for evaluating the migration job before you run it against the production database.

Ensure that you have met all of the prerequisites and completed the required preparations described in Preparing for Database Migration before you begin the migration procedures in this topic.
In addition be sure the following tasks are done:

- Obtain the necessary access credentials required.

  If Oracle Cloud Infrastructure Object Storage is used as the backup medium, obtain the Object Storage access credential. The user ID for the Oracle Cloud Infrastructure Console user and an auth token for Object Storage is required. If you are not using an existing auth token, a new auth token can be generated using the Oracle Cloud Infrastructure Console.

  If the source database server is accessed with the root user, then you need the root user password. If the source and target database serves are accessed with a private key file, then you need the private key file. The SYS password for the source database environment is also required.

  If Zero Data Loss Recovery Appliance is used as the backup medium, get the Zero Data Loss Recovery Appliance virtual private catalog (VPC) user credentials.

- Prepare the Zero Downtime Migration response file.

  The database migration is driven by a response file that captures the essential parameters for accomplishing the task.

  Use the sample RSP templates in `$ZDM_HOME/rhp/zdm/template/` file for example entries needed to set up the response file for your particular source, target, and backup environments.

The ZDMCLI `migrate database` command has options to let you test the migration before you run it in production. The options are highlighted in the following syntax examples.

ZDMCLI `migrate database` syntax for an Autonomous Database migration:

```
zdmuser> $ZDM_HOME/bin/zdmcli migrate database
 -rsp file_path
 -sourcedb source_db_unique_name_value
 -sourcenode host
 -srcauth zdmauth
 -srcarg1 user:username
 -srcarg2 identity_file:ssh_key_path
 -srcarg3 sudo_location:sudo_path
 -eval [-advisor [-ignoreadvisor] | -skipadvisor]]
```

ZDMCLI `migrate database` syntax for a co-managed database migration:

```
zdmuser> $ZDM_HOME/bin/zdmcli migrate database
 -rsp file_path
 -sourcedb source_db_unique_name_value
 -sourcenode host
 -srcauth zdmauth
 -srcarg1 user:username
 -srcarg2 identity_file:ssh_key_path
 -srcarg3 sudo_location:sudo_path
 -targetnode host
 -tgtauth zdmauth
 -tgtarg1 user:username
 -tgtarg2 identity_file:ssh_key_path
 -tgtarg3 sudo_location:sudo_path
 -eval [-advisor [-ignoreadvisor] | -skipadvisor]]
```

See the following topics for information about using the ZDMCLI `migrate database` options to evaluate your migration.

Using the ZDMCLI MIGRATE DATABASE -eval Option

Before submitting the database migration job for the production database, perform a test migration to determine how the process may fare with your configuration and settings.

It is highly recommended that for each migration you run `migrate database` in evaluation mode first. This evaluation allows you to correct any potential problems in the setup and configuration before performing the actual migration on a production database.

In evaluation mode, the migration process runs without effecting the changes. It is safe to run the command with the `-eval` option as many times as needed before running the actual migration job.

The command result output indicates the job ID for the evaluation migration job, which you can use to query the status of the job.

To run an evaluation of the migration process, run the ZDMCLI command `migrate database` with the `-eval` option, as shown in the following example.

Log in to the Zero Downtime Migration service host and switch to the `zdmuser` installed user.

```
su - zdmuser
```

If connectivity to the source database server is done through root credentials then the command would be the following:

```
zdmuser> $ZDM_HOME/bin/zdmcli migrate database -sourcedb
source_db_unique_name_value
-sourcenode source_database_server_name -srcroot
-targetnode target_database_server_name
-backupuser Object_store_login_user_name
-rsp response_file_location
-tgtauth zdmauth
-tgtarg1 user:target_database_server_login_user_name
-tgtarg2 identity_file:ZDM_installed_user_private_key_file_location
-tgtarg3 sudo_location:/usr/bin/sudo -eval
```

For the prompts, specify the source database SYS password and the source database server root user password. If the backup destination is Object Store (Bucket), then specify user swift authentication token. If the backup destination is Storage Classic (Container) then specify your tenancy login password.

For example,

```
zdmuser> $ZDM_HOME/bin/zdmcli migrate database -sourcedb zdmsdb -sourcenode
ocicdb1
-srcroot -targetnode ocidb1 -backupuser backup_user@example.com
-rsp /u01/app/zdmhome/rhp/zdm/template/zdm_template_zdmsdb.rsp -tgtauth
zdmauth
-tgtarg1 user:opc -tgtarg2 identity_file:/home/zdmuser/.ssh/
zdm_service_host.ppk -tgtarg3
sudo_location:/usr/bin/sudo -eval

Enter source database zdmsdb SYS password:
Enter source user "root" password:
Enter user "backup_user@example.com" password:
```

If connectivity to the source database server is through SSH key, then the command would be:

```
zdmuser> $ZDM_HOME/bin/zdmcli migrate database -sourcedb
source_db_unique_name_value
-sourcenode source_database_server_name -srcauth zdmauth
-srcarg1 user:source_database_server_login_user_name
-srcarg2 identity_file:ZDM_installed_user_private_key_file_location
-srcarg3 sudo_location:/usr/bin/sudo -targetnode target_database_server_name
-backupuser Object_store_login_user_name -rsp response_file_location
-tgtauth zdmauth -tgtarg1 user:target_database_server_login_user_name
-tgtarg2 identity_file:ZDM_installed_user_private_key_file_location
-tgtarg3 sudo_location:/usr/bin/sudo -eval
```

For the prompts, specify the source database SYS password. If the backup destination is Object Store (Bucket), then specify user swift authentication token. If the backup destination is Storage Classic (Container), then specify your tenancy login password.

```
zdmuser> $ZDM_HOME/bin/zdmcli migrate database -sourcedb zdmsdb -
sourcenode ocicdb1 -srcauth zdmauth
-srcarg1 user:opc -srcarg2 identity_file:/home/zdmuser/.ssh/
zdm_service_host.ppk
-srcarg3 sudo_location:/usr/bin/sudo -targetnode ocidb1 -backupuser
backup_user@example.com
-rsp /u01/app/zdmhome/rhp/zdm/template/zdm_template_zdmsdb.rsp -
tgtauth zdmauth -tgtarg1 user:opc
-tgtarg2 identity_file:/home/zdmuser/.ssh/zdm_service_host.ppk -
tgtarg3 sudo_location:/usr/bin/sudo -eval

Enter source database zdmsdb SYS password:
Enter user "backup_user@example.com" password:
```

Note that if a source single instance database is deployed without a Grid Infrastructure home, then in the above command use `-sourcesid` in place of `-sourcedb`.

Also, if a source database is configured for a `PASSWORD` based wallet, then add the `-tdekeystorepasswd` option to the command above, and for the prompt, specify the source database TDE keystore password value.

Note that the `-backupuser` argument takes the Object Storage access user or Zero Data Loss Recovery Appliance VPC user, and is skipped if NFS is the backup medium. For NFS, the source database user should have 'rwx' access to the NFS path provided.

The migration command checks for patch compatibility between the source and target home patch level, and expects the target home patch level to be equal to or higher than the source. If the target home patch level is not as expected, then the migration job is stopped and missing patches are reported. You can either patch the target home with the necessary patches or you can force continue the migration by appending the `-ignore PATCH_CHECK` or `-ignore ALL` option to the migration command.

The command result output indicates the job ID for the migration job, which you can use to query the status of the job.

If you want to run the command without providing passwords at the command line, see Provide Passwords Non-Interactively Using a Wallet.

Using the Cloud Premigration Advisor Tool

The Cloud Premigration Advisor Tool (CPAT) is a tool that performs analysis of the source database, looking for uses of database features and constructs that are problematic when migrating to one of Oracle's Autonomous Cloud offerings.

CPAT integrated with Zero Downtime Migration, and can be used with logical migration jobs. CPAT provides the following benefits:

- Warns you about any features used by you database that aren't supported in the target cloud environment

- Makes suggestions for remedial changes and/or parameters to use for the Data Pump export and import operations

When you run a logical migration using `ZDMCLI MIGRATE DATABASE`, CPAT is run by default as phase `ZDM_PRE_MIGRATION_ADVISOR`.

The following are options you can use with `ZDMCLI MIGRATE DATABASE` to customize how CPAT runs or to skip the CPAT phase.

- **`advisor`** only runs the minimum phases required for exclusively running Cloud Premigration Advisor Tool (CPAT) on the migration.
- **`ignoreadvisor`** ignores problems or errors reported by CPAT
- **`skipadvisor`** skips the CPAT phase in a migration job

See migrate database for more information about the command options, and see Cloud Premigration Advisor Tool (CPAT) Analyzes Databases for Suitability of Cloud Migration (Doc ID 2758371.1) for more information about CPAT.

Migrate the Database

Perform the database migration with Zero Downtime Migration using the `ZDMCLI migrate database` command.

Ensure that you have met all of the prerequisites and completed the required preparations described in Preparing for Database Migration before you begin the migration procedures in this topic.

In particular be sure the following tasks are done:

- Obtain the necessary access credentials required.

  If Oracle Cloud Infrastructure Object Storage is used as the backup medium, obtain the Object Storage access credential. The user ID for the Oracle Cloud Infrastructure Console user and an auth token for Object Storage is required. If you are not using an existing auth token, a new auth token can be generated using the Oracle Cloud Infrastructure Console.

  If the source database server is accessed with the root user, then you need the root user password. If the source and target database serves are accessed with a private key file, then you need the private key file. The SYS password for the source database environment is also required.

  If Zero Data Loss Recovery Appliance is used as the backup medium, get the Zero Data Loss Recovery Appliance virtual private catalog (VPC) user credentials.

- Prepare the Zero Downtime Migration response file.

  The database migration is driven by a response file that captures the essential parameters for accomplishing the task.

  Use the sample $ZDM_HOME/rhp/zdm/template/zdm_template.rsp file for example entries needed to set up the response file for your particular source, target, and backup environments.

- Determine if the migration process needs to be paused and resumed before you start the database migration. Once the migration job is initiated the job system runs the job as configured.

If the migration job needs to pause and resume at a particular point, then see the topics List Migration Job Phases and Pause and Resume Migration Job (cross references below) for more details.

Physical Migration with Root Credentials

The database migration job is submitted from the Zero Downtime Migration service host by the `zdmuser` user using the ZDMCLI command `migrate database`.

For a physical migration, if connectivity to the source database server is through root credentials, then the command would be:

```
zdmuser> $ZDM_HOME/bin/zdmcli migrate database
 -sourcedb source_db_unique_name_value
 -sourcenode source_database_server_name
 -srcroot
 -targetnode target_database_server_name
 -backupuser Object_store_login_user_name
 -rsp response_file_location
 -tgtauth zdmauth
 -tgtarg1 user:target_database_server_login_user_name
 -tgtarg2 identity_file:ZDM_installed_user_private_key_file_location
 -tgtarg3 sudo_location:/usr/bin/sudo
```

For the prompts, specify the source database SYS password and source database server root user password. If the backup destination is Object Store (Bucket), then specify user swift authentication token. If the backup destination is Storage Classic (Container), then specify your tenancy login password.

For example:

```
zdmuser> $ZDM_HOME/bin/zdmcli migrate database
 -sourcedb zdmsdb
 -sourcenode ocicdb1
 -srcroot
 -targetnode ocidb1
 -backupuser backup_user@example.com
 -rsp /u01/app/zdmhome/rhp/zdm/template/zdm_template_zdmsdb.rsp
 -tgtauth zdmauth
 -tgtarg1 user:opc
 -tgtarg2 identity_file:/home/zdmuser/.ssh/zdm_service_host.ppk
 -tgtarg3 sudo_location:/usr/bin/sudo

Enter source database zdmsdb SYS password:
Enter source user "root" password:
Enter user "backup_user@example.com" password:
```

Physical Migration with SSH Key

For a **physical migration**, if connectivity to the source database server is through **SSH key**, then the command would be:

```
zdmuser> $ZDM_HOME/bin/zdmcli migrate database
 -sourcedb source_db_unique_name_value
 -sourcenode source_database_server_name
 -srcauth zdmauth
 -srcarg1 user:source_database_server_login_user_name
 -srcarg2 identity_file:ZDM_installed_user_private_key_file_location
 -srcarg3 sudo_location:/usr/bin/sudo
 -targetnode target_database_server_name
 -backupuser Object_store_login_user_name
 -rsp response_file_location
 -tgtauth zdmauth
 -tgtarg1 user:target_database_server_login_user_name
 -tgtarg2 identity_file:ZDM_installed_user_private_key_file_location
 -tgtarg3 sudo_location:/usr/bin/sudo
```

For the prompts, specify the source database SYS password. If the backup destination is Object Store (Bucket), then specify user swift authentication token. If the backup destination is Storage Classic (Container), then specify your tenancy login password.

For example,

```
zdmuser> $ZDM_HOME/bin/zdmcli migrate database
 -sourcedb zdmsdb
 -sourcenode ocicdb1
 -srcauth zdmauth
 -srcarg1 user:opc
 -srcarg2 identity_file:/home/zdmuser/.ssh/zdm_service_host.ppk
 -srcarg3 sudo_location:/usr/bin/sudo
 -targetnode ocidb1
 -backupuser backup_user@example.com
 -rsp /u01/app/zdmhome/rhp/zdm/template/zdm_template_zdmsdb.rsp
 -tgtauth zdmauth
 -tgtarg1 user:opc
 -tgtarg2 identity_file:/home/zdmuser/.ssh/zdm_service_host.ppk
 -tgtarg3 sudo_location:/usr/bin/sudo

Enter source database zdmsdb SYS password:
Enter user "backup_user@example.com" password:
```

If a source single instance is deployed without a Grid Infrastructure home, then in the command above use `-sourcesid` in place of `-sourcedb`.

If the source database is configured for a `PASSWORD` based wallet, then add the `-tdekeystorepasswd` option to the command above, and for the prompt, specify the source database TDE keystore password value.

Note that the `-backupuser` argument takes the Object Storage access user or Zero Data Loss Recovery Appliance VPC user and is skipped if NFS is the backup medium. For NFS, the source database user should have 'rwx' access to the NFS path provided.

Logical Migration to Autonomous Database

For a **logical migration to Autonomous Database**, the command would be:

```
zdmuser> $ZDM_HOME/bin/zdmcli migrate database
 -rsp file_path
 -sourcedb source_db_unique_name_value
 -sourcenode host
 -srcauth zdmauth
 -srcarg1 user:username
 -srcarg2 identity_file:ssh_key_path
 -srcarg3 sudo_location:sudo_path
 -eval [-advisor [-ignoreadvisor] | -skipadvisor]]
```

Logical Migration to a Co-Managed Database

For a **logical migration to a co-managed system**, the command would be:

```
zdmuser> $ZDM_HOME/bin/zdmcli migrate database
 -rsp file_path
 -sourcedb source_db_unique_name_value
 -sourcenode host
 -srcauth zdmauth
 -srcarg1 user:username
 -srcarg2 identity_file:ssh_key_path
 -srcarg3 sudo_location:sudo_path
 -targetnode host
 -tgtauth zdmauth
 -tgtarg1 user:username
 -tgtarg2 identity_file:ssh_key_path
 -tgtarg3 sudo_location:sudo_path
 [-ignoreadvisor | -skipadvisor]
```

Patch Compatibility

The migration command checks for patch compatibility between the source and target home patch level, and expects the target home patch level to be equal to or higher than the source. If the target home patch level is not as expected, then the migration job is stopped and missing patches are reported. You can either patch the target home with the necessary patches or you can force continue the migration by appending the `-ignore PATCH_CHECK` or `-ignore ALL` option to the migration command.

Job ID Value

The command result output indicates the job ID for the migration job, which you can use to query the status of the job.

Running Migrations Non-Interactively

If you want to run the command without providing passwords at the command line, see Provide Passwords Non-Interactively Using a Wallet.

Query Migration Job Status

You can query the migration job status while the job is running.

Query the status of a database migration job using the ZDMCLI `query job` command, specifying the job ID. The job ID is shown in the command output when the database migration job is submitted.

```
zdmuser> $ZDM_HOME/bin/zdmcli query job -jobid job-id
```

You can find the console output of the migration job in the file indicated (Result file path:) in the `query job` command output. You can see migration progress messages in the specified file

List Migration Job Phases

You can list the operation phases involved in the migration job.

To list the operation phases involved in the migration job, add the `-listphases` option in the ZDMCLI `migrate` command. This option will list the phases involved in the operation.

For example,

```
zdmuser> $ZDM_HOME/bin/zdmcli migrate database -sourcedb zdmsdb -sourcenode
ocicdb1 -srcauth zdmauth
-srcarg1 user:opc -srcarg2 identity_file:/home/zdmuser/.ssh/
zdm_service_host.ppk -srcarg3 sudo_location:/usr/bin/sudo
-targetnode ocidb1 -backupuser backup_user@example.com -rsp /u01/app/
zdmhome/rhp/zdm/template/zdm_template_zdmsdb.rsp
-tgtauth zdmauth -tgtarg1 user:opc -tgtarg2 identity_file:/home/zdmuser/.ssh/
zdm_service_host.ppk
-tgtarg3 sudo_location:/usr/bin/sudo -listphases
```

Pause and Resume a Migration Job

You can pause a migration job at any point after the `ZDM_SETUP_TGT` phase, and resume the job at any time.

To pause a migration job, specify the `-pauseafter` option in the `ZDMCLI migrate` command with a valid phase to be paused after.

In the following example, if you specify `-pauseafter ZDM_SETUP_TGT`, the migration job will pause after completing the `ZDM_SETUP_TGT` phase.

```
zdmuser> $ZDM_HOME/bin/zdmcli migrate database -sourcedb zdmsdb -sourcenode
ocicdb1
-srcauth zdmauth -srcarg1 user:opc
-srcarg2 identity_file:/home/zdmuser/.ssh/zdm_service_host.ppk
-srcarg3 sudo_location:/usr/bin/sudo -targetnode ocidb1
-backupuser backup_user@example.com -rsp /u01/app/zdmhome/rhp/zdm/template/
zdm_template_zdmsdb.rsp -tgtauth zdmauth
-tgtarg1 user:opc -tgtarg2 identity_file:/home/zdmuser/.ssh/
zdm_service_host.ppk
-tgtarg3 sudo_location:/usr/bin/sudo -pauseafter ZDM_SETUP_TGT
```

Choosing a Migration Job Phase to Pause After

Choose a valid phase after `ZDM_CONFIGURE_DG_SRC` that is listed in the `migrate database ... -listphases` command output.

Note that the `-pauseafter` option allows only one phase to be specified.

The following are some cases where you may need to pause a migration job, and resume after some manual steps:

- Convert the Standby Database to TDE. If the source is not encrypted, of if ZDLRA is used as the data transfer medium, or the Oracle Database version is earlier than 12.2, you need to encrypt the target Cloud database.
- Enable Active Data Guard (optional)
- Monitor Data Guard Configuration Health, before the switchover
- Test the Cloud database (optional). You can convert the standby to the primary without doing the application switchover, which can be used to duplicate the source database for testing in the Cloud.

Best Practices for Pausing a Physical Migration Job

In a physical migration, if you use `-pauseafter` at phase `ZDM_CONFIGURE_DG_SRC`, then at the end of the execution of the phase, a standby is created at the target database and synchronization occurs between source and target databases.

Pausing the migration job after `ZDM_CONFIGURE_DG_SRC` is recommended so that you can do the following:

- Perform the application switchover to the cloud
- Manually encrypt the cloud database if ZDLRA is the backup method or your database release is earlier than Oracle Database 12.2
- Perform a failover to test the cloud database without making changes to the on-premises database

Preserving Log Files During a Paused Migration Job

To prevent source and target database log files from getting cleaned up between pausing and resuming a migration job, log files are written to `$ORACLE_BASE/zdm/zdm_`*`db_unique_name`*`_`*`zdm_job_id`*`/zdm/log` in their respective source and target database servers.

Ensure that all of the archive logs generated during and after the `ZDM_BACKUP_INCREMENTAL_SRC` phase are available, preferably until switchover, or at least until the target and source databases are in sync. Older archive logs, the archive logs generated prior to `ZDM_BACKUP_INCREMENTAL_SRC` are not needed.

Resuming a Migration Job

A paused job can be resumed any time by running the `ZDMCLI resume job` command, specifying the respective job ID.

```
zdmuser> $ZDM_HOME/bin/zdmcli resume job -jobid Job_ID
[-pauseafter valid-phase]
```

To schedule another pause, specify the `-pauseafter` option in the `resume` command with a valid phase to be paused after. Choose a valid phase later than phase currently paused at, that is listed in the `migrate database ... -listphases` command output.

Suspend and Resume a Migration Job

You can suspend a migration job at any point, and resume the job at any time.

To suspend a migration job run the following command.

```
zdmuser> $ZDM_HOME/bin/zdmcli suspend job -jobid job_id
```

Resuming a Suspended Migration Job

A suspended job can be resumed any time by running the `ZDMCLI resume job` command, specifying the respective job ID.

```
zdmuser> $ZDM_HOME/bin/zdmcli resume job -jobid Job_ID
[-pauseafter valid-phase]
```

You can optionally schedule a pause, by specifying the `-pauseafter` option in the `resume` command with a valid phase to be paused after. Choose a valid phase later than phase currently paused at, that is listed in the `migrate database ... -listphases` command output.

Rerun a Migration Job

If there are any unexpected errors in the migration workflow, you can correct them and rerun the migration job.

The errors are recorded in the job output, which can be queried using the ZDMCLI `query job` command. Upon resolving the error, the failed job can be continued from the point of failure.

Rerun the migration job by running the ZDMCLI `resume job` command, specifying the job ID of the job to be rerun, as shown here.

```
zdmuser> $ZDM_HOME/bin/zdmcli resume job -jobid Job_ID
```

Terminate a Running Migration Job

If you want to resubmit a database migration job for a specified database, you must first terminate the running migration job.

Zero Downtime Migration blocks attempts to rerun the `MIGRATE DATABASE` command for a specified database if that database is already part of an ongoing migration job.

If you want to resubmit a database migration job for a specified database, you must first terminate the running migration job in either `EXECUTING` or `PAUSED` state using the `ZDMCLI ABORT JOB` command.

```
zdmuser> $ZDM_HOME/bin/zdmcli abort job -jobid job-id
```

Zero Downtime Migration Centralized Fleet Migration Management

Zero Downtime Migration allows you to centrally monitor fleet level migration using the Object Storage Service PAR URL.

Zero Downtime centralized fleet migration management gives you the following capabilities.

- Fleet level migration monitoring
- Fleet level log aggregation for easy troubleshooting and data mining
- Centralized migration per job metrics collection, which can be leveraged for executive-level migration status dash-boards and future migration forecasting

Centralized fleet migration management also lets you to prohibit the operation team from accessing the source or target database server for migration failures. The operation team can be allowed to troubleshoot the failure with the logs available in the specified `ZDM_LOG_OSS_PAR_URL` or logs further staged to a different Oracle Cloud storage bucket for operation troubleshooting.

Centralized fleet migration management is only supported for physical migration jobs, and is enabled by setting the parameter `ZDM_LOG_OSS_PAR_URL` to a pre-authenticated URL, for example:

https://objectstorage.us-*region*.oraclecloud.com/ … /DEV_ZDM_LOGS_*RGN*/o/

Zero Downtime Migration uploads the job specific data, shown in the table below, to the specified OSS PAR URL at regular interval while a migration job is in progress.

Table 4-1 Centralized Fleet Migration Job Data

Category	URL Name space Suffix (Appended to value of ZDM_LOG_OSS_PAR_URL)	Content
Job Metrics	*PAR_URL*/1/*POD_NAME*/*ZDM_HOST*/*JOB_ID*/METRICS.txt For example, `/1/PRD_TEST_MIGRATION/s16izp/1/METRICS.txt`	Job Metrics details the source and target database vitals and backup and restore statistics per phase involved in the work flow and error. This data can be used for fleet level migration statistics for executive dashboards.
Job Status	*PAR_URL*/1*POD_NAME*/*ZDM_HOST*/*JOB_ID*/CURRENT_PHASE.txt For example, `/1/PRD_TEST_MIGRATION/s16izp/1/CURRENT_PHASE.txt`	Status entries are listed in the format *Phase Name*:*Phase Status* For example, `ZDM_BACKUP_FULL_SRC:EXECUTING` Phase Status values are `EXECUTING`, `PAUSED`, `FAILED`, or `COMPLETED`
Phase Log	*PAR_URL*/2/*TIMESTAMP*/*POD_NAME*/*ZDM_HOST*/*JOB_ID*/*ACTION HOST*/*PHASE_NAME*.log For example, 2/2011-11-07T17:58:34.049000/PRD_TEST_MIGRATION/2-zdm-01/1/cldx01/ZDM_BACKUP_FULL_SRC.log	Zero Downtime Migration uploads the phase-specific logs at the end of each phase with a UTC time-stamped name space. On job rerun, the time stamp would change and the rerun-specific log would be identified by latest time stamp.

Table 4-1 (Cont.) Centralized Fleet Migration Job Data

Category	URL Name space Suffix (Appended to value of ZDM_LOG_OSS_PAR_URL)	Content

Phase Status	`*PAR_URL*/1/*POD_NAME*/*ZDM_HOST*/*JOB_ID*/*PHASE_NAME*.txt`
	For example,
	`/1/PRD_TEST_MIGRATION/s16izp/1/ZDM_VALIDATE_TGT.txt`
		The file contents will be `COMPLETED` if the phase is completed or `PENDING` if it has yet to start.
		At the start of a migration job, Zero Downtime Migration creates all of the phase-named files with `PENDING` status, then updates them to `COMPLETED` as each phase is completed.
		Use the creation time of *`PHASE_NAME`*`.txt` to measure the phase elapsed time.

Progress Message	`*PAR_URL*/1/*POD_NAME*/*ZDM_HOST*/*JOB_ID*/CONSOLE.out`
	For example,
	`1/PRD_TEST_MIGRATION/s16izp/1/CONSOLE.out`
		Zero Downtime Migration updates `CONSOLE.out` for each *n* chunk of console messages received (for example, every 10 lines) normally, or instantly in case of errors.
		Progress messages are formatted like this:
		`node name : UTC Time : Progress message`
		For example, `hxvdbfz03: 2021-02-10T09:51:11.851Z : Retrieving information from source node "hfzb06" ...`

5

Managing the Zero Downtime Migration Service

Perform Zero Downtime Migration service life cycle operations using `zdmservice`.

Starting and Stopping the Zero Downtime Migration Service

You must start the Zero Downtime Migration service before you can migrate your databases using Zero Downtime Migration.

Start the Zero Downtime Migration service, `zdmservice`, as user `zdmuser`, with the following command.

```
zdmuser> $ZDM_HOME/bin/zdmservice start
```

If you must stop the Zero Downtime Migration service, run the following command.

```
zdmuser> $ZDM_HOME/bin/zdmservice stop
```

Checking Zero Downtime Migration Service Status

Check the status of the Zero Downtime Migration to see if it is running, and other service details.

To check the Zero Downtime Migration service status use the following command.

```
zdmuser> $ZDM_HOME/bin/zdmservice status
---------------------------------------
        Service Status
---------------------------------------
Running:        true
Tranferport:    5000-7000
Conn String:    jdbc:mysql://localhost:8897/
RMI port:       8895
HTTP port:      8896
Wallet path:    /u01/app/zdmbase/crsdata/fopds/security
```

Updating Zero Downtime Migration Software

If you already have Zero Downtime Migration software installed on a host, you can update it to the latest release. Zero Downtime Migration software updates give you the latest fixes while retaining existing job information, metadata, and log files.

Before you begin the software update, review the following requirements.

- Updating from Zero Downtime Migration 19c to 21c can only be done from the latest Zero Downtime Migration 19c software kit. Verify your 19c software kit using the following command:

  ```
  zdmuser> $ZDM_HOME/bin/zdmcli -build

  full version: 19.8.0.0.0
  label date: 200907
  ZDM kit build date: Thu Sep 24 06:22:07 PDT 2020
  ```

- Verify that your existing Zero Downtime Migration software install location has at least 15GB free space.
- Verify that you have enough space to back up the existing Zero Downtime Migration home (`ZDM_HOME`) and `ZDM_BASE` to the software download location.
- **Important**: Run the update script from outside of the currently installed Zero Downtime Migration home.

  Running the script from within a Zero Downtime Migration home results in home install and uninstall failures and leaves the service in an inconsistent state.
- The path specified in `ziploc` should have read/write access for `zdmuser`.
- All of the commands in the following procedure should be run as the existing Zero Downtime Migration software owner. For example, run as `zdmuser` in the examples that follow.

1. Download the Zero Downtime Migration software kit from https://www.oracle.com/database/technologies/rac/zdm-downloads.html to the Zero Downtime Migration service host.
2. Change to the directory to where Zero Downtime Migration software is downloaded and unzip the software.

   ```
   zdmuser> cd zdm_download_directory
   zdmuser> unzip zdmversion.zip
   ```

3. Run the `zdminstall.sh` script as the exiting Zero Downtime Migration home owner to update the software from the software download location.

   ```
   zdmuser>./zdminstall.sh update oraclehome=existing_zdm_oracle_home
      ziploc=zdm_software_location -zdm
   ```

   - `zmdinstall.sh` is the installation and update script
   - `oraclehome` is the absolute path to the Oracle Home where the existing Zero Downtime Migration software is installed
   - `ziploc` is the location of the compressed software file (zip) included in the Zero Downtime Migration kit

   For example,

   ```
   zdmuser>/u01/app/oracle/zdm/shiphome/update/zdminstall.sh update
        oraclehome=/u01/app/zdmhome
        ziploc=/u01/app/oracle/zdm/shiphome/update/zdm_home.zip -zdm
   ```

The update script does the following operations.

a. Backs up the existing Zero Downtime Migration home (`ZDM_HOME`) and `ZDM_BASE` into software download location

b. Stops the currently running Zero Downtime Migration service

c. Removes the currently installed Zero Downtime Migration home

d. Installs the new binaries in the Zero Downtime Migration home

e. Restores the configuration data.

The new Zero Downtime Migration home will retain all of the details of any migrations run with the previous Zero Downtime Migration home.

4. The Zero Downtime Migration service must be manually started after the upgrade. Start the Zero Downtime Migration service as user `zdmuser`.

```
zdmuser> $ZDM_HOME/bin/zdmservice start
```

You must start `zdmservice` before you can migrate your databases using Zero Downtime Migration.

If you must stop the Zero Downtime Migration service, run the following command.

```
zdmuser> $ZDM_HOME/bin/zdmservice stop
```

5. Verify that the Zero Downtime Migration service installation is successful.

When you run the following command, the output should be similar to that shown here.

```
zdmuser> $ZDM_HOME/bin/zdmservice status
---------------------------------------
        Service Status
---------------------------------------
Running:       true
Tranferport:   5000-7000
Conn String:   jdbc:mysql://localhost:8897/
RMI port:      8895
HTTP port:     8896
Wallet path:   /u01/app/zdmbase/crsdata/fopds/security
```

Uninstalling Zero Downtime Migration Software

Remove Zero Downtime Migration software from the Zero Downtime Migration service host.

All commands are run as `zdmuser`.

1. Stop the Zero Downtime Migration service.

```
zdmuser> $ZDM_HOME/bin/zdmservice stop
```

2. Run the following command to uninstall the software.

```
zdmuser> $ZDM_HOME/bin/zdmservice deinstall
```

Performing a Silent Update or Deinstallation

You can skip the confirmation prompt during Zero Downtime Migration software update or deinstallation to ensure these operations run smoothly.

A `-silent` option is available for zdminstall.sh `update` or `deinstall` operations to avoid being asked for confirmation, as shown in the following examples.

Example of silent update:

```
zdmuser> cd zdm_download_directory
zdmuser> unzip zdmversion.zip
...
zdmuser>./zdminstall.sh update -silent
oraclehome=absolute_path_to_zdm_home
ziploc=zdm_software_location -zdm
```

Example of silent deinstall:

```
zdmuser> $ZDM_HOME/bin/zdmservice deinstall -silent
```

Setting the MySQL Port

You can discover and set the port number that Zero Downtime Services uses for MySQL.

MySQL Default Port Number

Zero Downtime Migration uses MySQL internally, configuring it by default on port 3306. If a port number is not specified and the default is not available, Zero Downtime Migration increases the port value by one and retries up to five times.

Finding the Current Port Number

Run `zdmservice status` to see the current MySQL port number in the connection string, as shown here.

```
zdmuser> $ZDM_HOME/bin/zdmservice status

Conn String: jdbc:mysql://localhost:8897/
```

Changing the Port Number

You can change this default to another value using the `zdmservice modify mysqlPort=`*`port`* option.

```
zdmuser> $ZDM_HOME/bin/zdmservice modify mysqlPort=port
```

6 Troubleshooting Zero Downtime Migration

This section describes how to handle migration job failures.

For more information about troubleshooting Zero Downtime Migration and known issues in the current release, see the Zero Downtime Migration Release Notes.

Handling Migration Job Failures

If your migration job fails, the following logs can help you discover the issue.

Migration Job Output Logs

If your migration job encounters an error, refer to the migration job output logs, Zero Downtime Migration service logs, and server-specific operational phase logs present at the respective source or target database servers.

If the migration job encounters an exception (that is, fails) then the logs can provide some indication of the nature of the fault. The logs for the migration procedures executed in the source and target environments are stored on the servers in the respective source and target environments. The Zero Downtime Migration command output location is provided to you when the migration job is run with the `ZDMCLI` command `migrate database`. You can also find the log file location (`Result file path`) in the output of the `ZDMCLI` command `query job -jobid` *`job-id`*.

Zero Downtime Migration Service Host Log

Determine which operational phase the migration job was in at the time of failure, and whether the phase belongs to the source (phase name contains `SRC`) or target (phase name contains `TGT`) . Check the Zero Downtime Migration service host log at `$ZDM_BASE/crsdata/`*`zdm_service_host`*`/rhp/zdmserver.log.0`, and access the respective source or target server to check the log associated with the operational phase in `$ORACLE_BASE/zdm/zdm_`*`db_unique_name`*`_`*`job-id`*`/zdm/log`.

If the Zero Downtime Migration service does not start, then check the Zero Downtime Migration service logs for process start-up errors to determine the cause of the error reported. The Zero Downtime Migration service log can be found at `$ZDM_BASE/crsdata/`*`zdm_service_host`*`/rhp/zdmserver.log.0`.

Data Pump Log

For logical migration jobs on co-managed target databases, refer to the Data Pump log in the specified `DATA_PUMP_DIR`. For Autonomous Database targets, the Data Pump logs are uploaded to a specified Object Storage bucket. If the data bucket is not specified, then you will need to upload the dump from Autonomous Database to Object Storage to access it. See How To View Import Log Generated For ADW/ATP (Doc ID 2448060.1)

If a migration job fails, you can fix the cause of the failure and then re-run the job while monitoring the logs for progress.

A

Zero Downtime Migration Port Requirements

The ports required for communication between the Zero Downtime Migration service host, the source and target database servers, and Oracle Cloud Object Store Service are described in the following table.

Table A-1 Zero Downtime Migration Communication Ports

Initiator	Target	Protocol	Port	Purpose	Description
Zero Downtime Migration service host	Source and target database servers	TCP	22	SSH	Authentication-based operations to run Zero Downtime Migration operational phases Source and target database servers should accept incoming connections from the Zero Downtime Migration service host. Not applicable to Autonomous Database targets
Zero Downtime Migration service host	Source and target database servers	TCP	1521, 2484, or a database SCAN Listener port applicable for job	SQL*Net	For logical migrations
Zero Downtime Migration service host	Oracle Cloud Interface REST endpoint	SSL	443	OCI REST endpoint	Target discovery for logical migrations

Table A-1 (Cont.) Zero Downtime Migration Communication Ports

Initiator	Target	Protocol	Port	Purpose	Description
Source database servers	Target database servers	TCP	1521 or database SCAN Listener port applicable for the job	SQL*Net	Should allow Oracle client connections to the database over Oracle's SQL*Net protocol Perform database queries, Data Guard sync, and configuration **Note:** If you are using a non-default port number (that is, something other than port 1521) for the local listener address, then the non-default port should allow connections.

Table A-1 (Cont.) Zero Downtime Migration Communication Ports

Initiator	Target	Protocol	Port	Purpose	Description
Target database servers	Source database servers	TCP	1521 or a database SCAN Listener port applicable	SQL*Net	Should allow Oracle client connections to the database over Oracle's SQL*Net protocol Allows redo log shipping if source database needs to be in sync with the new primary on Oracle Cloud after switchover. If there is no communication possible from Oracle Cloud to source database server then set `SKIP_FALLBACK` to `TRUE` in the response file to avoid this communication. **Note:** If you are using a non-default port number (that is, something other than port 1521) for the local listener address, then the non-default port should allow connections.
Source database servers	Oracle Cloud Object Store Service	SSL	443	Database backup store. Create a backup of the source database to the specified Oracle Cloud Object store container.	If the chosen backup method uses Oracle Cloud Object Store Service as the backup medium, then access ports as documented Oracle Cloud Object Store Service applies.

Table A-1 (Cont.) Zero Downtime Migration Communication Ports

Initiator	Target	Protocol	Port	Purpose	Description
Target database servers	Oracle Cloud Object Store Service	SSL	443	Database backup store. Restore backup of the source database from the specified Oracle Cloud Object store container to the target database.	If the chosen backup method uses Oracle Cloud Object Store Service as the backup medium, then access ports as documented Oracle Cloud Object Store Service applies.

Note:

When performing a migration with root credentials (`migrate database -scroot`), during the setup phase, Zero Downtime Migration uses six ports chosen from the ephemeral range, or six ports from the range of ports set in `TRANSFERPORT_RANGE` in *`zdmbase`*`/crsdata/`*`zdm_service_host`*`/rhp/conf/rhp.pref`. The specified ports must be allowed to accept incoming connections from the Zero Downtime Migration service host on the source or target database server.

B Zero Downtime Migration Encryption Requirements

Zero Downtime Migration does not always require encryption at the source (although, all Cloud databases are encrypted by default). The following tables list specific cases when encryption is not required.

Table B-1 On-Premises Unencrypted Primary and Cloud Encrypted Standby

Operation	On-Premises Primary 11g R2	Cloud Standby 11g R2	On-Premises Primary 12c R1	Cloud Standby 12c R1	On-Premises Primary 12c R2	Cloud Standby 12c R2 and later	Notes
Data Guard initial setup for on-premises primary and cloud standby	Unencrypted	Encrypted	Unencrypted	Encrypted	Unencrypted	Encrypted	In these cases the standby database is manually encrypted after instantiation
New tablespace creation on-premises primary	Unencrypted	Unencrypted	Unencrypted	Unencrypted	Unencrypted	Unencrypted	Requires manual TDE conversion for standby database
Redo generated in on-premises primary	Unencrypted	Unencrypted	Unencrypted	Unencrypted	Unencrypted	Unencrypted	
Archived logs	Unencrypted	Unencrypted	Unencrypted	Unencrypted	Unencrypted	Unencrypted	
New and changed blocks	Unencrypted	Encrypted	Unencrypted	Encrypted	Unencrypted	Encrypted	Redo shipped from the on-premises primary to the cloud is not encrypted
Recovery in the cloud standby	N/A	Encrypted	N/A	Encrypted	N/A	Encrypted	Redo shipped from the on-premises primary to the cloud is not encrypted

Table B-2 Cloud Encrypted Primary and On-Premises Unencrypted Standby

Operation	Cloud Primary 11g R2	On-Premises Standby 11g R2	Cloud Primary 12c R1	On-Premises Standby 12c R1	Cloud Primary 12c R2	On-Premises Standby 12c R2 and later	Notes
New tablespace creation in cloud primary	Encrypted	Encrypted	Encrypted	Encrypted	Encrypted	Encrypted	ASO required for on-premises to decrypt
Redo generated in cloud primary	Encrypted	Encrypted	Encrypted	Encrypted	Encrypted	Encrypted	ASO required for on-premises to decrypt
Archived logs	Encrypted	Encrypted	Encrypted	Encrypted	Encrypted	Encrypted	ASO required for on-premises to decrypt
New and changed blocks for existing unencrypted tablespace on standby	Encrypted	Encrypted*	Encrypted	Encrypted*	Encrypted	Unencrypted	ASO is required on-premises to decrypt and encrypt * For 11g R2 and 12c R1 redo apply will encrypt only if redo is encrypted
Recovery in the on-premises standby	N/A	Encrypted	N/A	Encrypted	N/A	Unencrypted data depends on whether the datafile is encrypted	ASO required for on-premises database

C

Provide Passwords Non-Interactively Using a Wallet

You can avoid entering passwords in the command line and run the `ZDMCLI MIGRATE DATABASE` command without user interaction.

> **Note:**
>
> This wallet-based password input option is only supported for physical online and offline migration jobs.

Currently, whenever you submit the `$ZDM_HOME/bin/zdmcli migrate database` command, it prompts for the source database `SYS` password, Object Store user swift authentication token, and the source database Transparent Data Encryption (TDE) keystore password (if the wallet was configured as a `PASSWORD`-based TDE wallet). If you don't want to be required to enter the password at the command line, such as when you do automation using Rundeck, complete the following steps.

Run the following commands on the Zero Downtime Migration service host as Zero Downtime Migration software owner (for example, `zdmuser`).

1. Create an auto-login wallet for the source database `SYS` user.
   a. Create a directory where you want to create and store the wallet.

      ```
      zdmuser> mkdir sys_wallet_path
      ```

      For example:

      ```
      /u01/app/zdmhome> mkdir sysWallet
      ```

   b. Create a wallet.

      ```
      zdmuser> $ZDM_HOME/bin/orapki wallet create -wallet sys_wallet_path
      -auto_login_only
      ```

      For example

      ```
      /u01/app/zdmhome> $ZDM_HOME/bin/orapki wallet create -wallet
      sysWallet
      -auto_login_only
      Oracle PKI Tool Release 19.0.0.0.0 - Production
      Version 19.4.0.0.0
      Copyright (c) 2004, 2019, Oracle and/or its affiliates. All rights
      reserved.
      ```

```
Operation is successfully completed.
```

c. Add a `SYS` user login credentials to wallet.

```
zdmuser> $ZDM_HOME/bin/mkstore -wrl sys_wallet_path
-createCredential store sysuser
```

At the prompt, enter the source database `SYS` password.

For example

```
/u01/app/zdmhome> $ZDM_HOME/bin/mkstore -wrl ./sysWallet
-createCredential store sysuser
Oracle Secret Store Tool Release 19.0.0.0.0 - Production
Version 19.4.0.0.0
Copyright (c) 2004, 2019, Oracle and/or its affiliates. All
rights reserved.

Your secret/Password is missing in the command line
Enter your secret/Password:
Re-enter your secret/Password:
```

d. Verify that the wallet files were created.

```
zdmuser> ls -l sys_wallet_path
```

For example

```
/u01/app/zdmhome> ls -l sysWallet/
total 4
-rw-------. 1 opc opc 581 Jun  2 08:00 cwallet.sso
-rw-------. 1 opc opc   0 Jun  2 08:00 cwallet.sso.lck
```

2. Create an auto-login wallet for the Object Store user.

   a. Create a directory where you want to create and store the wallet.

```
zdmuser> mkdir oss_wallet_path
```

For example

```
/u01/app/zdmhome> mkdir ossWallet
```

   b. Create a wallet

```
zdmuser> $ZDM_HOME/bin/orapki wallet create -wallet
oss_wallet_path
-auto_login_only
```

For example

```
/u01/app/zdmhome> $ZDM_HOME/bin/orapki wallet create
-wallet ./ossWallet -auto_login_only
Oracle PKI Tool Release 19.0.0.0.0 -Production
Version 19.4.0.0.0
Copyright (c) 2004, 2019,
Oracle and/or its affiliates. All rights reserved.

 Operation is successfully completed.
```

c. Add the Object Store user login credentials to the wallet.

```
zdmuser> $ZDM_HOME/bin/mkstore -wrl oss_wallet_path
-createCredential store ossuser
```

For the prompt,

- If the backup destination is Object Store (Bucket), then enter the user swift authentication token.
- If the backup destination is Storage Classic (Container), then enter your tenancy login password.

For example

```
/u01/app/zdmhome> $ZDM_HOME/bin/mkstore -wrl ./ossWallet
-createCredential store ossuser
Oracle Secret Store Tool Release 19.0.0.0.0 - Production
Version 19.4.0.0.0
Copyright (c) 2004, 2019, Oracle and/or its affiliates. All rights
reserved.

Your secret/Password is missing in the command line
Enter your secret/Password:
Re-enter your secret/Password:
```

d. Verify that the wallet files were created.

```
zdmuser> ls -l oss_wallet_path
```

For example

```
/u01/app/zdmhome> ls -l ./ossWallet
total 4
-rw-------. 1 opc opc 597 Jun  2 08:02 cwallet.sso
-rw-------. 1 opc opc   0 Jun  2 08:01 cwallet.sso.lck
```

3. Create an auto-login wallet for the source database TDE keystore.

a. Create a directory where you want to create and store the wallet.

```
zdmuser> mkdir tde_wallet_path
```

For example

```
/u01/app/zdmhome> mkdir tdeWallet
```

b. Create a wallet.

```
zdmuser> $ZDM_HOME/bin/orapki wallet create -wallet
tde_wallet_path
-auto_login_only
```

For example

```
/u01/app/zdmhome> $ZDM_HOME/bin/orapki wallet create -wallet ./
tdeWallet
-auto_login_only
Oracle PKI Tool Release 19.0.0.0.0 - Production
Version 19.4.0.0.0
Copyright (c) 2004, 2019, Oracle and/or its affiliates. All
rights reserved.

Operation is successfully completed.
```

c. Add the source database TDE keystore credentials to the wallet.

```
zdmuser> $ZDM_HOME/bin/mkstore -wrl tde_wallet_path
-createCredential store tdeuser
```

At the prompt, enter the TDE keystore password.

For example

```
/u01/app/zdmhome> $ZDM_HOME/bin/mkstore -wrl ./tdeWallet
-createCredential store tdeuser
Oracle Secret Store Tool Release 19.0.0.0.0 - Production
Version 19.4.0.0.0
Copyright (c) 2004, 2019, Oracle and/or its affiliates. All
rights reserved.

Your secret/Password is missing in the command line
Enter your secret/Password:
Re-enter your secret/Password:
```

d. Verify that the wallet files were created.

```
zdmuser> ls -l tde_wallet_path
```

For example

```
/u01/app/zdmhome> ls -l tdeWallet
total 4
-rw-------. 1 opc opc 581 Jun  2 08:06 cwallet.sso
-rw-------. 1 opc opc   0 Jun  2 08:04 cwallet.sso.lck
```

Setting Command Options to Access the Wallets

To specify wallet information in the `ZDMCLI MIGRATE DATABASE` command, set the `-sourcesyswallet`, `-osswallet`, and `-tdekeystorewallet` options as shown here.

```
zdmuser> $ZDM_HOME/bin/zdmcli migrate database
-sourcedb source_db_unique_name_value
-sourcenode source_database_server_name -srcauth zdmauth
-srcarg1 user:source_database_server_login_user_name
-srcarg2 identity_file:zdm_installed_user_private_key_file_location
-srcarg3 sudo_location:/usr/bin/sudo -targetnode target_database_server_name
-backupuser object_store_login_user_name -rsp response_file_location
-tgtauth zdmauth -tgtarg1 user:target_database_server_login_user_name
-tgtarg2 identity_file:zdm_installed_user_private_key_file_location
-tgtarg3 sudo_location:/usr/bin/sudo -sourcesyswallet sys_wallet_path
-osswallet oss_wallet_path  -tdekeystorewallet tde_wallet_path
-eval
```

- `-sourcesyswallet` *`sys_wallet_path`* specifies the full path for the auto-login wallet file on the Zero Downtime Migration host containing the `SYS` password of the source database
- `-osswallet` *`oss_wallet_path`* specifies the full path for the auto-login wallet file on the Zero Downtime Migration host containing credentials for the Object Storage Service backup user
- `-tdekeystorewallet` *`tde_wallet_path`* specifies the full path for the auto-login wallet file on the Zero Downtime Migration host containing the TDE keystore password

Evaluation Mode Example

```
zdmuser> $ZDM_HOME/bin/zdmcli migrate database -sourcedb zdmsdb -sourcenode
ocicdb1
-srcauth zdmauth -srcarg1 user:opc
-srcarg2 identity_file:/home/zdmuser/.ssh/zdm_service_host.ppk
-srcarg3 sudo_location:/usr/bin/sudo -targetnode ocidb1
-backupuser backup_user@example.com
-rsp /u01/app/zdmhome/rhp/zdm/template/zdm_template_zdmsdb.rsp -tgtauth
zdmauth
-tgtarg1 user:opc -tgtarg2 identity_file:/home/zdmuser/.ssh/
zdm_service_host.ppk
-tgtarg3 sudo_location:/usr/bin/sudo -sourcesyswallet /u01/app/zdmhome/
sysWallet
-osswallet /u01/app/zdmhome/ossWallet -eval

Operation "zdmcli migrate database" scheduled with the job ID "1".
```

Migration Mode Example

```
zdmuser> $ZDM_HOME/bin/zdmcli migrate database -sourcedb zdmsdb -sourcenode
ocicdb1
-srcauth zdmauth -srcarg1 user:opc
-srcarg2 identity_file:/home/zdmuser/.ssh/zdm_service_host.ppk
-srcarg3 sudo_location:/usr/bin/sudo -targetnode ocidb1
-backupuser backup_user@example.com
```

```
-rsp /u01/app/zdmhome/rhp/zdm/template/zdm_template_zdmsdb.rsp -
tgtauth zdmauth
-tgtarg1 user:opc -tgtarg2 identity_file:/home/zdmuser/.ssh/
zdm_service_host.ppk
-tgtarg3 sudo_location:/usr/bin/sudo -sourcesyswallet /u01/app/zdmhome/
sysWallet
-osswallet /u01/app/zdmhome/ossWallet

Operation "zdmcli migrate database" scheduled with the job ID "2".
```

D

Zero Downtime Migration Process Phases

The migration job process in Zero Downtime Migration runs in operational phases as a work flow. The tables below describe the phases for physical and logical migrations.

Example D-1 Listing Zero Downtime Migration Process Phases

Run the ZDMCLI `migrate database` command with the `-listphases` option to list the operational phases for your migration job, as shown here.

```
zdmuser> $ZDM_HOME/bin/zdmcli migrate database -sourcedb zdmsdb
-sourcenode ocicdb1 -srcauth zdmauth -srcarg1 user:opc
-srcarg2 identity_file:/home/zdmuser/.ssh/zdm_service_host.ppk
-srcarg3 sudo_location:/usr/bin/sudo -targetnode ocidb1
-backupuser backup_user@example.com
-rsp /u01/app/zdmhome/rhp/zdm/template/zdm_template_zdmsdb.rsp
-tgtauth zdmauth -tgtarg1 user:opc
-tgtarg2 identity_file:/home/zdmuser/.ssh/zdm_service_host.ppk
-tgtarg3 sudo_location:/usr/bin/sudo -listphases
```

Table D-1 Physical Migration Phase Descriptions

Phase name	Online/Offline	Description
ZDM_GET_SRC_INFO	Both	Gets information about the source database
ZDM_GET_TGT_INFO	Both	Gets information about the target database
ZDM_SETUP_SRC	Both	Sets up Zero Downtime Migration helper modules on the source server
ZDM_SETUP_TGT	Both	Sets up Zero Downtime Migration helper modules on the target server
ZDM_PREUSERACTIONS	Both	Runs migration pre-user actions, if any, at the source
ZDM_PREUSERACTIONS_TGT	Both	Runs migration pre-user actions, if any, at the target
ZDM_VALIDATE_SRC	Both	Perform validations at the source
ZDM_VALIDATE_TGT	Both	Performs validations at the target
ZDM_OBC_INST_SRC	Both	Installs Oracle Database Cloud Backup Module at the source
ZDM_OBC_INST_TGT	Both	Installs Oracle Database Cloud Backup Module at the target
ZDM_BACKUP_FULL_SRC	Both	Performs full backup of the source database
ZDM_BACKUP_INCREMENTAL_SRC	Both	Performs incremental backup of the source database
ZDM_DISCOVER_SRC	Both	Performs database discovery at the source for setting up Data Guard

Table D-1 (Cont.) Physical Migration Phase Descriptions

Phase name	Online/Offline	Description
ZDM_COPYFILES	Both	Copies Oracle password file and TDE wallets from source to target
ZDM_SETUP_TDE_TGT	Both	Copies TDE wallet files from the source to the target keystore location
ZDM_OSS_RESTORE_TGT	Offline	Performs full database restore
ZDM_BACKUP_DIFFERENTIAL_SRC	Offline	Performs differential backup of the source database
ZDM_OSS_RECOVER_TGT	Offline	Performs incremental restore, recovery database, and opens database with reset logs
ZDM_PREPARE_TGT	Both	Prepares target for Data Guard standby creation
ZDM_CLONE_TGT	Online	Creates Data Guard standby from the Cloud backup
ZDM_FINALIZE_TGT	Both	Finalizes Data Guard standby preparation of the target. Converts the target database to RAC if originally provisioned as Oracle RAC.
ZDM_CONFIGURE_DG_SRC	Online	Registers the Cloud standby with the source
ZDM_SWITCHOVER_SRC	Online	Initiates switchover actions at the source
ZDM_SWITCHOVER_TGT	Online	Completes switchover actions at the target
ZDM_POST_DATABASE_OPEN_TGT	Both	Performs activities after database is opened, such as restore pluggable database state, DBA directories, RMAN configuration
ZDM_DATAPATCH_TGT	Both	Runs datapatch at the target
ZDM_SHUTDOWN_SRC	Both	Shuts down source database at the end of the migration
ZDM_POSTUSERACTIONS	Both	Performs any post-migration user actions at the source
ZDM_POSTUSERACTIONS_TGT	Both	Performs any post-migration user actions at the target
ZDM_CLEANUP_SRC	Both	Performs clean up at the source
ZDM_CLEANUP_TGT	Both	Performs clean up at the target

Table D-2 Logical Migration Phase Descriptions

Phase name	Online/Offline	Description
ZDM_VALIDATE_SRC	Both	Validates the source database access credentials, database parameter settings. Online only: verifies `ggadmin` user privileges and GoldenGate capture support for objects in source database.

Table D-2 (Cont.) Logical Migration Phase Descriptions

Phase name	Online/Offline	Description
ZDM_VALIDATE_TGT	Both	Verifies that the target database exists, discovers the database type, and validates access credentials, security, and connectivity. For an Autonomous Database target, sets up the access to the target database from the Zero Downtime Migration node by downloading the target database access wallet with OCI REST services and discovers the target OCPU. Online only: verifies `ggadmin` user privileges.
ZDM_PRE_MIGRATION_ADVISOR	Both	Runs the Oracle Database Premigration Advisor Tool on the migration job.
ZDM_VALIDATE_DATAPUMP_SETTINGS_SRC	Both	Validates the export directory object (if applicable), and checks for sufficient space and permission for specified user in the source database to export dumps. Checks if the specified Oracle Cloud Object Store buckets, data bucket, and wallet bucket are accessible from the source. Also validates the proxy configuration if applicable.
ZDM_VALIDATE_DATAPUMP_SETTINGS_TGT	Both	Verifies that the Data Pump import directory object exists. If a pre-existing DBLINK was specified, checks if it exists and is valid, and ensures that the Autonomous Database requirements for the DBLINK and wallet files are met. For a co-managed target database, ensures that the OCI OSS data bucket and wallet bucket are valid and accessible from the target server. Ensures that the local path to download the dump is valid and has sufficient space.
ZDM_VALIDATE GG_HUB	Online	Verifies GoldenGate Microservices REST endpoints, software configuration, health, and connectivity to the source and target databases.
ZDM_PREPARE_GG_HUB	Online	Registers database connection details and credentials with GoldenGate Microservices.
ZDM_ADD_HEARTBEAT_SRC	Online	Creates GoldenGate heartbeat table in the source database. If the table already exists, sets update frequency to 60 seconds.
ZDM_ADD_SCHEMA_TRANDATA_SRC	Online	Prepares the source database schemas for instantiation by enabling schema level supplemental logging.
ZDM_CREATE_GG_EXTRACT_SRC	Online	Starts the GoldenGate Extract process at the source database
ZDM_PREPARE_DATAPUMP_SRC	Both	Creates a new directory object for Data Pump, if required. Creates OCI Auth Token to access OCI OSS bucket if required.

Table D-2 (Cont.) Logical Migration Phase Descriptions

Phase name	Online/Offline	Description
ZDM_PREPARE_DATAPUMP_TGT	Both	Creates a new directory object for Data Pump, if required. Stores OCI Auth token in the database for secure OSS access. If migrating via DBLINK, and a DBLINK must be created, creates the necessary database credentials to access the source and create a new DBLINK. Ensures Autonomous Database security requirements are met using DBLINK over SSL.
ZDM_DATAPUMP_EXPORT_SRC	Both	Starts and monitors the Data Pump Export on the source database.
ZDM_UPLOAD_DUMPS_SRC	Both	Uploads Data Pump dump files from the source to OCI OSS.
ZDM_DATAPUMP_IMPORT_TGT	Both	Starts import of Data Pump Dumps to the target database, either from the OCI OSS bucket or via DBLINK, and monitors the Data Pump import progress.
ZDM_POST_DATAPUMP_SRC	Both	Removes any Data Pump directory object created by Zero Downtime Migration.
ZDM_POST_DATAPUMP_TGT	Both	Fixes any invalid objects in the target database. Removes the database access and OCI OSS access credentials that were created for the migration. Removes any DBLINK created by Zero Downtime Migration. Optionally, removes source database dumps stored in OCI OSS bucket.
ZDM_ADD_HEARTBEAT_TGT	Online	Creates the GoldenGate heartbeat table in the target database. If the table already exists, sets update frequency to 60 seconds.
ZDM_ADD_CHECKPOINT_TGT	Online	Creates GoldenGate checkpoint table in the target database to track Replicat progress.
ZDM_CREATE_GG_REPLICAT_TGT	Online	Starts GoldenGate Replicat process for the target database.
ZDM_MONITOR_GG_LAG	Online	Polls the GoldenGate checkpoint and heartbeat data to measure end-to-end apply lag until lag decreases below desired threshold.
ZDM_SWITCHOVER_APP	Online	If the source database is idle, stops GoldenGate Extract, waits for GoldenGate Replicat to complete apply, and stops GoldenGate Replicat.
ZDM_RM_GG_EXTRACT_SRC	Online	Deletes GoldenGate Extract process on source database
ZDM_RM_GG_REPLICAT_TGT	Online	Deletes GoldenGate Replicat process on target database
ZDM_DELETE_SCHEMA_TRANDATA_SRC	Online	Disables schema level supplemental logging on source database

Table D-2   (Cont.) Logical Migration Phase Descriptions

Phase name	Online/Offline	Description
ZDM_RM_HEARTBEAT_SRC	Online	Drops the GoldenGate heartbeat table in source database, if the table was created by Zero Downtime Migration. Otherwise, resets update frequency to original setting.
ZDM_RM_CHECKPOINT_TGT	Online	Drops the GoldenGate checkpoint table in the target database.
ZDM_RM_HEARTBEAT_TGT	Online	Drops the GoldenGate heartbeat table in the target database, if the table was created by Zero Downtime Migration. Otherwise, resets the update frequency to the original value.
ZDM_CLEAN_GG_HUB	Online	Deletes the database connection details and credentials saved with GoldenGate Microservices
ZDM_POST_ACTIONS	Both	Removes Autonomous Database access wallet from the Zero Downtime Migration node.

E

Oracle Data Pump Settings for Zero Downtime Migration

Setting Advanced Data Pump Parameters

You might want to select specific schemas to migrate, rename tablespaces, or include or exclude specific objects from the as part of a migration.

The following are example parameter settings you can use to specify these selections or changes when you set `DATAPUMPSETTINGS_JOBMODE=FULL` or `DATAPUMPSETTINGS_JOBMODE=SCHEMA` job modes.

These parameters are set in the response file at `$ZDM_HOME/rhp/zdm/template/zdm_logical_template.rsp`.

To exclude specific object types:

```
DATAPUMPSETTINGS_DATAPUMPPARAMETERS_EXCLUDETYPELIST=COMMENT,DOMAIN_INDEX,MATERIA
LIZED_VIEW_LOG,RLS_POLICY,TRIGGER
```

To exclude select SCHEMA objects for DATAPUMPSETTINGS_JOBMODE=FULL mode:

```
DATAPUMPSETTINGS_METADATAFILTERS-1=name:NAME_EXPR,value:'NOT
IN(''SYSMAN'')',objectType:SCHEMA
```

```
DATAPUMPSETTINGS_METADATAFILTERS-3=name:NAME_EXPR,value:'NOT
IN(''SH'')',objectType:SCHEMA
```

Note:

The `SCHEMA` name `SYSMAN` is surrounded by two single quotes and not a double quote.

To exclude select SCHEMA objects for DATAPUMPSETTINGS_JOBMODE=SCHEMA mode:

```
EXCLUDEOBJECTS-1=owner:SYSMAN
```

```
EXCLUDEOBJECTS-2=owner:SCOTT
```

By default, Zero Downtime Migration ignores Oracle Maintained Objects.

To include select SCHEMA objects for DATAPUMPSETTINGS_JOBMODE=SCHEMA mode:

```
INCLUDEOBJECTS-1=owner:SYSMAN
```

```
INCLUDEOBJECTS-2=owner:SCOTT
```

By default, Zero Downtime Migration ignores Oracle Maintained Objects.

To REMAP the tablespaces:

```
DATAPUMPSETTINGS_METADATAREMAPS-1=type:REMAP_TABLESPACE,oldValue:TS_DATA_X,newVa
lue:DATA
```

```
DATAPUMPSETTINGS_METADATAREMAPS-2=type:REMAP_TABLESPACE,oldValue:DBS,newVa
lue:DATA
```

Data Pump Parameter Default Settings

Zero Downtime Migration automatically sets optimal defaults for Data Pump parameters to achieve better performance and ensure security of data. The following table lists the Data Pump parameters set by Zero Downtime Migration, and the values they are set to.

If there is a Zero Downtime Migration response file parameter available to override the default, it is listed in the Optional Zero Downtime Migration Response File Parameter to Override column. The override parameters are set in the response file at `$ZDM_HOME/rhp/zdm/template/zdm_logical_template.rsp`.

Table E-1 Data Pump Parameter Defaults

Data Pump Parameter	Default Value	Optional ZDM Response File Parameter to Override
EXCLUDE	index (ADW-S) cluster (ADB-D, ADB-S) indextype (ADW-S) materialized_view (ADW-S) materialized_view_log (ADW-S) materialized_zonemap (ADW-S) db_link (ADB) statistics (User managed Target and ADB)	Allows additional `EXCLUDE` entries to be specified **Note** Specifying invalid object types for `EXCLUDE` will lead to a Data Pump export error. Ensure that a valid object type is specified for the `DATAPUMPSETTINGS_DATAPUMPPARAMETERS_EXCLUDETYPELIST` parameter. To see a list of valid object types, query the following views: `DATABASE_EXPORT_OBJECTS` for `FULL` mode, `SCHEMA_EXPORT_OBJECTS` for `SCHEMA` mode, and `TABLE_EXPORT_OBJECTS` for `TABLE` and `TABLESPACE` mode. The values listed in the `OBJECT_PATH` column are the valid object types. For example, specifying the invalid object type parameter in the response file will lead to export error. `ORA-39038: Object path "<specified invalid>" is not supported for SCHEMA jobs.`

Table E-1 (Cont.) Data Pump Parameter Defaults

Data Pump Parameter	Default Value	Optional ZDM Response File Parameter to Override
PARALLEL	ZDM sets PARALLEL parameter by default as follows For User managed DB :- (Sum of (2 x (no. of physical CPU) per node)) with Max 32 cap. For ADB :- No. of OCPUs	DATAPUMPSETTINGS_DATAPUMPPARAMETERS_IMPORTPARALLELISMDEGREE DATAPUMPSETTINGS_DATAPUMPPARAMETERS_EXPORTPARALLELISMDEGREE
CLUSTER	ZDM always sets the Cluster mode as default	DATAPUMPSETTINGS_DATAPUMPPARAMETERS_NOCLUSTER
COMPRESSION	COMPRESSION_ALGORITHM is set to BASIC(for 11.2) and MEDIUM (for 12.1+) COMPRESSION is set to ALL	N/A
ENCRYPTION	ENCRYPTION is set to ALL ENCRYPTION_ALGORITHM is set to AES128 ENCRYPTION_MODE is set to PASSWORD	N/A
FILESIZE	FILESIZE is set to 5G	N/A
FLASHBACK_SCN	For OFFLINE_LOGICAL ZDM set FLASHBACK_TIME System time now. For ONLINE LOGICAL ZDM uses neither FLASHBACK_SCN not FLASHBACK_TIME	N/A
REUSE_DUMPFILES	Always set to YES	N/A
TRANSFORM	Always sets OMIT_ENCRYPTION_CLAUSE:Y for 19c+ targets Always sets LOB_STORAGE:SECUREFILE For ADB target, following transform is set by default SEGMENT_ATTRIBUTES:N DWCS_CVT_IOTS:Y CONSTRAINT_USE_DEFAULT_INDEX:Y	Allows additional TRANSFORM to be specified
METRICS	Always set to Yes	N/A
LOGTIME	Always set to ALL	N/A
TRACE	Always set to 1FF0b00	N/A

Table E-1 (Cont.) Data Pump Parameter Defaults

Data Pump Parameter	Default Value	Optional ZDM Response File Parameter to Override
LOGFILE	Always set to Data Pump job name and created under specified export or import directory object. Say if Data Pump job is ZDM_2_DP_EXPORT_8417 and directory object used is DATA_PUMP_DIR, then the operation log is created by name ZDM_2_DP_EXPORT_8417.log under DATA_PUMP_DIR.	N/A

Data Pump Error Handling

Some errors are ignored by Zero Downtime Migration. You must review any remaining errors appearing in the Data Pump log.

The following Data Pump errors are ignored by Zero Downtime Migration.

- ORA-31684: XXXX already exists
- ORA-39111: Dependent object type XXXX skipped, base object type
- ORA-39082: Object type ALTER_PROCEDURE: XXXX created with compilation warnings

Ensure that you clear all Cloud Premigration Advisor Tool (CPAT) reported errors to avoid any underlying Data Pump errors.

F

Zero Downtime Migration Physical Migration Response File Parameters Reference

The following topics describe the Zero Downtime Migration response file parameters used in physical migrations.

BACKUP_PATH

`BACKUP_PATH` specifies a valid path accessible at the source and target for migration backup type.

Property	Description
Syntax	`BACKUP_PATH = {STORAGEPATH | EXTBACKUP}`
Default value	There is no default value.
Range of values	`STORAGEPATH` - NFS backup location `EXTBACKUP` - external backup location Leave this parameter value blank for other migration backup types

DATA_TRANSFER_MEDIUM

`DATA_TRANSFER_MEDIUM` specifies the media used for the source database backup. Valid values are dependent on whether `MIGRATION_METHOD=ONLINE_PHYSICAL` or `MIGRATION_METHOD=OFFLINE_PHYSICAL`.

Property	Description
Syntax	`DATA_TRANSFER_MEDIUM = {OSS | NFS | EXTBACKUP | ZDLRA}`
Default value	`OSS`

Property	Description
Range of values	When `MIGRATION_METHOD=ONLINE_PHYSICAL` You can use one of these data transfer method values: • `OSS` - Oracle Data Guard using OSS for standby initialization, supported for Oracle Cloud Infrastructure(OCI) virtual machine or bare metal, Exadata Cloud Service (EXACS) and Exadata Cloud at Customer (EXACC) • `NFS` - Oracle Data Guard using backup location such as NFS, supported for Exadata Cloud at Customer (EXACC) • `EXTBACKUP` - Oracle Data Guard with existing backup in external location, supported for Exadata Cloud at Customer (EXACC) • `ZDLRA` - Oracle Data Guard using ZDLRA for standby initialization, supported for Exadata Cloud at Customer (EXACC) When `MIGRATION_METHOD=OFFLINE_PHYSICAL` You can use one of these data transfer method values: • `OSS` - Migration using backup and restore through OSS, supported for Oracle Cloud Infrastructure(OCI) virtual machine or bare metal, Exadata Cloud Service (EXACS) and Exadata Cloud at Customer (EXACC) and no SQL*Net connectivity needed between source and target • `NFS` - Migration using backup and restore through NFS, supported for Exadata Cloud at Customer (EXACC) and no SQL*Net connectivity needed between source and target

DATAPATCH_WITH_ONE_INSTANCE_RUNNING

`DATAPATCH_WITH_ONE_INSTANCE_RUNNING` specifies whether or not to stop all instances except one running on the target database server when the datapatch utility is run. When datapatch completes all of the stopped instances are started.

Property	Description
Syntax	`DATAPATCH_WITH_ONE_INSTANCE_RUNNING ={TRUE | FALSE}`
Default value	`FALSE`
Range of values	`TRUE` - Stops all instances except one running on the target database server when running the datapatch utility. `FALSE` - Does not stop all instances except one running on the target database server when running the datapatch utility.

HOST

`HOST` specifies the cloud storage REST endpoint URL to access Oracle Cloud Object Storage.

Property	Description
Syntax	`HOST = ` *`rest_endpoint_url`*
Default value	There is no default value.
Range of values	For Oracle Cloud Infrastructure storage the typical value format is `https://swiftobjectstorage.us-phoenix-1.oraclecloud.com/v1/`*`ObjectStorageNamespace`* For Oracle Cloud Infrastructure Classic storage the typical value format is `https://acme.storage.oraclecloud.com/v1/Storage-`*`tenancy name`*

Usage Notes

To access Oracle Cloud Object Storage, you must set both the `HOST` and `OPC_CONTAINER` parameters.

MAX_DATAPATCH_DURATION_MINS

`MAX_DATAPATCH_DURATION_MINS` specifies a timeout value, in minutes, after which if the datapatch utility has failed to complete then the operation is stopped.

Property	Description
Syntax	`MAX_DATAPATCH_DURATION_MINS = ` *`minutes`*
Default value	There is no default value. Zero Downtime Migration waits until datapatch completes by default.

MIGRATION_METHOD

`MIGRATION_METHOD` specifies whether the migration uses Oracle Data Guard (online) or backup and restore (offline).

Property	Description
Syntax	`MIGRATION_METHOD = {ONLINE_PHYSICAL | OFFLINE_PHYSICAL}`
Default value	This is no default value

Property	Description
Range of values	`ONLINE_PHYSICAL` uses Data Guard switchover migration method `OFFLINE_PHYSICAL` uses backup and restore database migration method

NONCDBTOPDB_CONVERSION

`NONCDBTOPDB_CONVERSION` indicates whether to convert a source database from non-CDB to PDB or skip the conversion.

Property	Description
Syntax	`NONCDBTOPDB_CONVERSION = {TRUE | FALSE}`
Default value	`FALSE`
Range of values	`TRUE` performs the conversion `FALSE` does not perform the conversion

NONCDBTOPDB_SWITCHOVER

`NONCDBTOPDB_SWITCHOVER`, for a physical migration using Data Guard switchover, indicates whether the switchover operations will be executed during a migration job with non-CDB to PDB conversion enabled.

Property	Description
Syntax	`NONCDBTOPDB_SWITCHOVER = {TRUE | FALSE}`
Default value	`TRUE`
Range of values	`TRUE` performs the switchover `FALSE` does not perform the switchover

OPC_CONTAINER

`OPC_CONTAINER` specifies the Object Storage bucket (called the container on Oracle Cloud Infrastructure Classic), and is set to access Oracle Cloud Object Storage.

Property	Description
Syntax	`OPC_CONTAINER = `*`bucket`*
Default value	There is no default value.

Usage Notes

To access Oracle Cloud Object Storage, you must set both the `HOST` and `OPC_CONTAINER` parameters.

The bucket is also referred to as a container for Oracle Cloud Infrastructure Classic storage.

PLATFORM_TYPE

`PLATFORM_TYPE` specifies the target database platform.

Property	Description
Syntax	`PLATFORM_TYPE = {VMDB | EXACS | EXACC | NON_CLOUD}`
Default value	`VMDB`
Range of values	`VMDB` - indicates target platform is Oracle Cloud Infrastructure(OCI) virtual machine or bare metal `EXACS` - indicates target platform is Exadata Cloud Service `EXACC` - indicates target platform is Exadata Cloud at Customer `NON_CLOUD` - indicates the target is an on-premises environment

SHUTDOWN_SRC

`SHUTDOWN_SRC` specifies whether or not to shut down the source database after migration completes.

Property	Description
Syntax	`SHUTDOWN_SRC ={TRUE | FALSE}`
Default value	`FALSE`
Range of values	`TRUE` - Shut down the source database after migration completes. `FALSE` - Dos not shut down the source database after migration completes.

SKIP_FALLBACK

`SKIP_FALLBACK` specifies whether or not to ship redo logs from the primary (target) database to the standby (source) database, either voluntarily or because there is no connectivity between the target and source database servers.

Property	Description
Syntax	`SKIP_FALLBACK = {TRUE | FALSE}`
Default value	`FALSE`
Range of values	`TRUE` - do not ship redo logs from the primary (target) database to the standby (source) database. `FALSE` - ship redo logs from the primary (target) database to the standby (source) database.

SKIP_SRC_SERVICE_RETENTION

`SKIP_SRC_SERVICE_RETENTION` specifies whether or not to retain the source database services and run them on the target database. This parameter is only effective for the `BACKUP_RESTORE_OSS` and `BACKUP_RESTORE_NFS` migration methods.

Property	Description
Syntax	`SKIP_SRC_SERVICE_RETENTION ={TRUE | FALSE}`
Default value	`FALSE`
Range of values	`TRUE` - Do not retain the source database services. `FALSE` - Retain the source database services.

SRC_BASTION_HOST_IP

`SRC_BASTION_HOST_IP` specifies the bastion host IP address, if you want to connect to the source database server using a bastion host.

Property	Description
Syntax	`SRC_BASTION_HOST_IP =` *`IP_address`*
Default value	There is no default value.

Usage Notes

If you want to connect to the source database server using a bastion host, values for the bastion host IP address parameter, `SRC_BASTION_HOST_IP`, and the source database host IP address parameter, `SRC_HOST_IP`, are required in the Zero Downtime Migration response file.

If you do not want to use the default value, set the following parameters for bastion host connection.

`SRC_BASTION_PORT` - The port number defaults to 22 if not specified.

`SRC_BASTION_USER` - The bastion host source user is only required if the user specified for the source `zdmauth` plug-in is different from the user of the source bastion host. The bastion user defaults to the user specified for the source `zdmauth` plug-in if the parameter is not specified.

`SRC_BASTION_IDENTITY_FILE` - If not specified, the value defaults to the value specified for the `identity_file` argument of the source `zdmauth` plug-in.

SRC_BASTION_IDENTITY_FILE

`SRC_BASTION_IDENTITY_FILE` specifies the bastion user, if you want to connect to the source database server using a bastion host.

Property	Description
Syntax	`SRC_BASTION_IDENTITY_FILE = identity_file`
Default value	If not specified, the value defaults to the value specified for the `identity_file` argument of the source `zdmauth` plug-in.

Usage Notes

If you want to connect to the source database server using a bastion host, values for the bastion host IP address parameter, `SRC_BASTION_HOST_IP`, and the source database server IP address parameter, `SRC_HOST_IP`, are required in the Zero Downtime Migration response file.

If you do not want to use the default value, set the following parameters for bastion host connection.

`SRC_BASTION_PORT` - The port number defaults to 22 if not specified.

`SRC_BASTION_USER` - The bastion host source user is only required if the user specified for the source `zdmauth` plug-in is different from the user of the source bastion host. The bastion user defaults to the user specified for the source `zdmauth` plug-in if the parameter is not specified.

`SRC_BASTION_IDENTITY_FILE` - If not specified, the value defaults to the value specified for the `identity_file` argument of the source `zdmauth` plug-in.

SRC_BASTION_PORT

`SRC_BASTION_PORT` specifies the bastion host port number, if you want to connect to the source database server using a bastion host.

Property	Description
Syntax	`SRC_BASTION_PORT = port_number`
Default value	`22`

Usage Notes

If you want to connect to the source database server using a bastion host, values for the bastion host IP address parameter, `SRC_BASTION_HOST_IP`, and the source database server IP address parameter, `SRC_HOST_IP`, are required in the Zero Downtime Migration response file.

If you do not want to use the default value, set the following parameters for bastion host connection.

`SRC_BASTION_PORT` - The port number defaults to 22 if not specified.

`SRC_BASTION_USER` - The bastion host source user is only required if the user specified for the source `zdmauth` plug-in is different from the user of the source bastion host. The bastion user defaults to the user specified for the source `zdmauth` plug-in if the parameter is not specified.

SRC_BASTION_IDENTITY_FILE - If not specified, the value defaults to the value specified for the identity_file argument of the source zdmauth plug-in.

SRC_BASTION_USER

SRC_BASTION_USER specifies the bastion user, if you want to connect to the source database server using a bastion host.

Property	Description
Syntax	SRC_BASTION_USER = *bastion_user*
Default value	The bastion user defaults to the user specified for the source zdmauth plug-in if the parameter is not specified.

Usage Notes

If you want to connect to the source database server using a bastion host, values for the bastion host IP address parameter, SRC_BASTION_HOST_IP, and the source database server IP address parameter, SRC_HOST_IP, are required in the Zero Downtime Migration response file.

If you do not want to use the default value, set the following parameters for bastion host connection.

SRC_BASTION_PORT - The port number defaults to 22 if not specified.

SRC_BASTION_USER - The bastion host source user is only required if the user specified for the source zdmauth plug-in is different from the user of the source bastion host. The bastion user defaults to the user specified for the source zdmauth plug-in if the parameter is not specified.

SRC_BASTION_IDENTITY_FILE - If not specified, the value defaults to the value specified for the identity_file argument of the source zdmauth plug-in.

SRC_CONFIG_LOCATION

SRC_CONFIG_LOCATION specifies the SSH configuration file location on the Zero Downtime Migration service host (host where Zero Downtime Migration service is running).

Property	Description
Syntax	SRC_CONFIG_LOCATION = *SSH_config_file_path*
Default value	*User_home*/.ssh/config

Usage Notes

Set SRC_CONFIG_LOCATION to the full path of the SSH configuration file location on the Zero Downtime Migration service host, for example, /home/crsuser/.ssh/config.

SRC_DB_LISTENER_PORT

`SRC_DB_LISTENER_PORT` indicates the source database listener port. Set this property when there is Standalone Database (no Grid Infrastructure) configured with non-default SCAN listener port other than 1521.

Property	Description
Syntax	`SRC_DB_LISTENER_PORT =` *`listener_port_number`*
Default value	1521

SRC_HOST_IP

`SRC_HOST_IP` specifies the bastion user, if you want to connect to the source database server using a bastion host.

Property	Description
Syntax	`SRC_HOST_IP =` *`IP_address`*
Default value	There is no default value.

Usage Notes

If you want to connect to the source database server using a bastion host, values for the bastion host IP address parameter, `SRC_BASTION_HOST_IP`, and the source database server IP address parameter, `SRC_HOST_IP`, are required in the Zero Downtime Migration response file.

If you do not want to use the default value, set the following parameters for bastion host connection.

`SRC_BASTION_PORT` - The port number defaults to 22 if not specified.

`SRC_BASTION_USER` - The bastion host source user is only required if the user specified for the source `zdmauth` plug-in is different from the user of the source bastion host. The bastion user defaults to the user specified for the source `zdmauth` plug-in if the parameter is not specified.

`SRC_BASTION_IDENTITY_FILE` - If not specified, the value defaults to the value specified for the `identity_file` argument of the source `zdmauth` plug-in.

SRC_HTTP_PROXY_PORT

`SRC_HTTP_PROXY_PORT` specifies the HTTPS proxy port number on the source database server if an SSH connection needs to connect using a proxy.

Property	Description
Syntax	`SRC_HTTP_PROXY_PORT =` *`https_proxy_port_number`*
Default value	There is no default value.

SRC_HTTP_PROXY_URL

`SRC_HTTP_PROXY_URL` specifies the HTTPS proxy URL on the source database server if an SSH connection needs to connect using a proxy.

Property	Description
Syntax	`SRC_HTTP_PROXY_URL = `*`https_proxy_url`*
Default value	There is no default value.

SRC_OSS_PROXY_HOST

`SRC_OSS_PROXY_HOST` specifies the Object Storage Service proxy host on the source database server if a proxy is needed for connecting to the Object Store.

Property	Description
Syntax	`SRC_OSS_PROXY_HOST = `*`oss_proxy_host`*
Default value	There is no default value.

Usage Notes

Set both the `SRC_OSS_PROXY_HOST` and `SRC_OSS_PROXY_PORT` parameters if a proxy is needed for connecting to the Object Store.

SRC_OSS_PROXY_PORT

`SRC_OSS_PROXY_PORT` specifies the Object Storage Service proxy port number on the source database server if a proxy is needed for connecting to the Object Store.

Property	Description
Syntax	`SRC_OSS_PROXY_PORT = `*`oss_proxy_port_number`*
Default value	There is no default value.

Usage Notes

Set both the `SRC_OSS_PROXY_HOST` and `SRC_OSS_PROXY_PORT` parameters if a proxy is needed for connecting to the Object Store.

SRC_PDB_NAME

`SRC_PDB_NAME` indicates the source database PDB name.

Property	Description
Syntax	`SRC_PDB_NAME = `*`source_database_pdb_name`*

Property	Description
Default value	No default value

SRC_RMAN_CHANNELS

`SRC_RMAN_CHANNELS` specifies the number of RMAN channels to be allocated at the source database server for performing RMAN backups.

Property	Description
Syntax	`SRC_RMAN_CHANNELS =` *`number_of_channels`*
Default value	`10`

SRC_SSH_RETRY_TIMEOUT

`SRC_SSH_RETRY_TIMEOUT` specifies a timeout value, in minutes, after which Zero Downtime Migration stops attempting SSH connections after an initial failure to connect.

Property	Description
Syntax	`SRC_SSH_RETRY_TIMEOUT =` *`number_of_minutes`*
Default value	There is no default value.

SRC_TIMEZONE

`SRC_TIMEZONE` specifies the source database server time zone, which is needed for SIDB case when there is no Grid Infrastructure configured.

Property	Description
Syntax	`SRC_TIMEZONE =` *`source_db_time_zone`*
Default value	There is no default value.

SRC_ZDLRA_WALLET_LOC

`SRC_ZDLRA_WALLET_LOC` specifies the path of the Zero Data Loss Recovery Appliance wallet on the source database server.

Property	Description
Syntax	`SRC_ZDLRA_WALLET_LOC =` *`source_zdlra_wallet_location`* The expected format for the location is `/u02/app/oracle/product/12.1.0/dbhome_3/dbs/zdlra`
Default value	There is no default value.

Usage Notes

When using Zero Data Loss Recovery Appliance as the migration backup medium, you must set the following parameters.

`SRC_ZDLRA_WALLET_LOC`

`TGT_ZDLRA_WALLET_LOC`

`ZDLRA_CRED_ALIAS`

TGT_BASTION_HOST_IP

`TGT_BASTION_HOST_IP` specifies the bastion host IP address, if you want to connect to the target database server using a bastion host.

Property	Description
Syntax	`TGT_BASTION_HOST_IP =` *`bastion_ip_address`*
Default value	There is no default value.

Usage Notes

If you want to connect to the target database server using a bastion host, you are required to configure values for the bastion host IP address parameter, `TGT_BASTION_HOST_IP`, and the target database server IP address parameter, `TGT_HOST_IP`, in the Zero Downtime Migration response file.

If you do not want to use the default values for the remaining bastion connection parameters, set the following parameters to configure the bastion host connection.

`TGT_BASTION_PORT` - The port number defaults to 22 if not specified.

`TGT_BASTION_USER` - The bastion host target user is only required if the user specified for the target `zdmauth` plug-in is different from the user of the target bastion host. The bastion user defaults to the user specified for the target `zdmauth` plug-in if the parameter is not specified.

`TGT_BASTION_IDENTITY_FILE` - If not specified, the value defaults to the value specified for the `identity_file` argument of the target `zdmauth` plug-in.

TGT_BASTION_IDENTITY_FILE

`TGT_BASTION_IDENTITY_FILE` specifies the bastion user, if you want to connect to the target database server using a bastion host.

Property	Description
Syntax	`TGT_BASTION_IDENTITY_FILE =` *`identity_file`*
Default value	If not specified, the value defaults to the value specified for the `identity_file` argument of the target `zdmauth` plug-in.

Usage Notes

If you want to connect to the target database server using a bastion host, you are required to configure values for the bastion host IP address parameter, `TGT_BASTION_HOST_IP`, and the target database server IP address parameter, `TGT_HOST_IP`, in the Zero Downtime Migration response file.

If you do not want to use the default values for the remaining bastion connection parameters, set the following parameters to configure the bastion host connection.

`TGT_BASTION_PORT` - The port number defaults to 22 if not specified.

`TGT_BASTION_USER` - The bastion host target user is only required if the user specified for the target `zdmauth` plug-in is different from the user of the target bastion host. The bastion user defaults to the user specified for the target `zdmauth` plug-in if the parameter is not specified.

`TGT_BASTION_IDENTITY_FILE` - If not specified, the value defaults to the value specified for the `identity_file` argument of the target `zdmauth` plug-in.

TGT_BASTION_PORT

`TGT_BASTION_PORT` specifies the bastion host port number, if you want to connect to the target database server using a bastion host.

Property	Description
Syntax	`TGT_BASTION_PORT = *port_number*`
Default value	`22`

Usage Notes

If you want to connect to the target database server using a bastion host, you are required to configure values for the bastion host IP address parameter, `TGT_BASTION_HOST_IP`, and the target database server IP address parameter, `TGT_HOST_IP`, in the Zero Downtime Migration response file.

If you do not want to use the default values for the remaining bastion connection parameters, set the following parameters to configure the bastion host connection.

`TGT_BASTION_PORT` - The port number defaults to 22 if not specified.

`TGT_BASTION_USER` - The bastion host target user is only required if the user specified for the target `zdmauth` plug-in is different from the user of the target bastion host. The bastion user defaults to the user specified for the target `zdmauth` plug-in if the parameter is not specified.

`TGT_BASTION_IDENTITY_FILE` - If not specified, the value defaults to the value specified for the `identity_file` argument of the target `zdmauth` plug-in.

TGT_BASTION_USER

`TGT_BASTION_USER` specifies the bastion user, if you want to connect to the target database server using a bastion host.

Property	Description
Syntax	`TGT_BASTION_USER = bastion_user`
Default value	The bastion user defaults to the user specified for the target `zdmauth` plug-in if the parameter is not specified.

Usage Notes

If you want to connect to the target database server using a bastion host, you are required to configure values for the bastion host IP address parameter, `TGT_BASTION_HOST_IP`, and the target database server IP address parameter, `TGT_HOST_IP`, in the Zero Downtime Migration response file.

If you do not want to use the default values for the remaining bastion connection parameters, set the following parameters to configure the bastion host connection.

`TGT_BASTION_PORT` - The port number defaults to 22 if not specified.

`TGT_BASTION_USER` - The bastion host target user is only required if the user specified for the target `zdmauth` plug-in is different from the user of the target bastion host. The bastion user defaults to the user specified for the target `zdmauth` plug-in if the parameter is not specified.

`TGT_BASTION_IDENTITY_FILE` - If not specified, the value defaults to the value specified for the `identity_file` argument of the target `zdmauth` plug-in.

TGT_CONFIG_LOCATION

`TGT_CONFIG_LOCATION` specifies the SSH configuration file location on the Zero Downtime Migration service host (host where Zero Downtime Migration service is running).

Property	Description
Syntax	`TGT_CONFIG_LOCATION =` *`SSH_config_file_path`*
Default value	*`User_home`*`/.ssh/config`

Usage Notes

Set `TGT_CONFIG_LOCATION` to the full path of the SSH configuration file location on the Zero Downtime Migration service host, for example, `/home/crsuser/.ssh/config`.

TGT_DATAACFS

`TGT_DATAACFS` specifies the location for the data files ACFS volume (`data`) on the target database. Use only if required to override the values discovered automatically by Zero Downtime Migration.

Property	Description
Syntax	`TGT_DATAACFS = *data_location*`
Default value	There is no default value.

Usage Notes

Zero Downtime Migration discovers the location for ASM and ACFS `data`, `redo`, and `reco` storage volumes from the specified target database, making these target database storage properties optional.

If you need to override the values automatically discovered by Zero Downtime Migration, then you can set the following parameters. For example, `TGT_DATADG=+DATAC3`

For ASM use these parameters

`TGT_DATADG`

`TGT_REDODG`

`TGT_RECODG`

For ACFS use these parameters

`TGT_DATAACFS`

`TGT_REDOACFS`

`TGT_RECOACFS`

TGT_DATADG

`TGT_DATADG` specifies the location for the data files ASM disk group (`data`) on the target database. Use only if required to override the values discovered automatically by Zero Downtime Migration.

Property	Description
Syntax	`TGT_DATADG = *data_location*`
Default value	There is no default value.

Usage Notes

Zero Downtime Migration discovers the location for ASM and ACFS `data`, `redo`, and `reco` storage volumes from the specified target database, making these target database storage properties optional.

If you need to override the values automatically discovered by Zero Downtime Migration, then you can set the following parameters. For example, `TGT_DATADG=+DATAC3`

For ASM use these parameters

`TGT_DATADG`

`TGT_REDODG`

`TGT_RECODG`

For ACFS use these parameters

`TGT_DATAACFS`

`TGT_REDOACFS`

`TGT_RECOACFS`

TGT_DB_UNIQUE_NAME

`TGT_DB_UNIQUE_NAME` is used by Zero Downtime Migration to identify the target database.

Property	Description
Syntax	`TGT_DB_UNIQUE_NAME =` *`value of target database DB_UNIQUE_NAME`*
Default value	

Usage Notes

Set `TGT_DB_UNIQUE_NAME` to the target database `DB_UNIQUE_NAME` value.

If the target database is Oracle Cloud Infrastructure, Exadata Cloud Service, or Exadata Cloud at Customer, the target database `DB_UNIQUE_NAME` parameter value must be unique to ensure that Oracle Data Guard can identify the target as a different database from the source database.

TGT_HOST_IP

`TGT_HOST_IP` specifies the bastion user, if you want to connect to the target database server using a bastion host.

Property	Description
Syntax	`TGT_HOST_IP =` *`IP_address`*
Default value	There is no default value.

Usage Notes

If you want to connect to the target database server using a bastion host, you are required to configure values for the bastion host IP address parameter, `TGT_BASTION_HOST_IP`, and the target database server IP address parameter, `TGT_HOST_IP`, in the Zero Downtime Migration response file.

If you do not want to use the default values for the remaining bastion connection parameters, set the following parameters to configure the bastion host connection.

`TGT_BASTION_PORT` - The port number defaults to 22 if not specified.

`TGT_BASTION_USER` - The bastion host target user is only required if the user specified for the target `zdmauth` plug-in is different from the user of the target bastion host. The bastion user defaults to the user specified for the target `zdmauth` plug-in if the parameter is not specified.

`TGT_BASTION_IDENTITY_FILE` - If not specified, the value defaults to the value specified for the `identity_file` argument of the target `zdmauth` plug-in.

TGT_HTTP_PROXY_PORT

`TGT_HTTP_PROXY_PORT` specifies the HTTPS proxy port if an SSH connection needs to use a proxy to connect to the target database server.

Property	Description
Syntax	`TGT_HTTP_PROXY_PORT =` *`https_proxy_port_number`*
Default value	There is no default value.

Usage Notes

Set both the `TGT_HTTP_PROXY_URL` and `TGT_HTTP_PROXY_PORT` parameters if the SSH connection needs to use an HTTPS proxy to connect to the target database server.

TGT_HTTP_PROXY_URL

`TGT_HTTP_PROXY_URL` specifies the HTTPS proxy URL if an SSH connection needs to use a proxy to connect to the target database server.

Property	Description
Syntax	`TGT_HTTP_PROXY_URL =` *`https_proxy_url`*
Default value	There is no default value.

Usage Notes

Set both the `TGT_HTTP_PROXY_URL` and `TGT_HTTP_PROXY_PORT` parameters if the SSH connection needs to use an HTTPS proxy to connect to the target database server.

TGT_OSS_PROXY_HOST

`TGT_OSS_PROXY_HOST` specifies the Object Storage Service proxy host on the target database server if a proxy is needed for connecting to the Object Store.

Property	Description
Syntax	`TGT_OSS_PROXY_HOST =` *`oss_proxy_host`*
Default value	There is no default value.

Usage Notes

Set both the `TGT_OSS_PROXY_HOST` and `TGT_OSS_PROXY_PORT` parameters if a proxy is needed for connecting to the Object Store.

TGT_OSS_PROXY_PORT

`TGT_OSS_PROXY_PORT` specifies the Object Storage Service proxy port number on the target database server if a proxy is needed for connecting to the Object Store.

Property	Description
Syntax	`TGT_OSS_PROXY_PORT =` *`oss_proxy_port_number`*
Default value	There is no default value.

Usage Notes

Set both the `TGT_OSS_PROXY_HOST` and `TGT_OSS_PROXY_PORT` parameters if a proxy is needed for connecting to the Object Store.

TGT_RECOACFS

`TGT_RECOACFS` specifies the location for the fast recovery area ACFS volume (`reco`) on the target database. Use only if required to override the values discovered automatically by Zero Downtime Migration.

Property	Description
Syntax	`TGT_RECOACFS =` *`reco_location`*
Default value	There is no default value.

Usage Notes

Zero Downtime Migration discovers the location for ASM and ACFS `data`, `redo`, and `reco` storage volumes from the specified target database, making these target database storage properties optional.

If you need to override the values automatically discovered by Zero Downtime Migration, then you can set the following parameters. For example, `TGT_DATADG=+DATAC3`

For ASM use these parameters

`TGT_DATADG`

`TGT_REDODG`

`TGT_RECODG`

For ACFS use these parameters

`TGT_DATAACFS`

`TGT_REDOACFS`

`TGT_RECOACFS`

TGT_RECODG

`TGT_RECODG` specifies the location for the fast recovery area ASM disk group (`reco`) on the target database. Use only if required to override the values discovered automatically by Zero Downtime Migration.

Property	Description
Syntax	`TGT_RECODG = ` *`reco_location`*
Default value	There is no default value.

Usage Notes

Zero Downtime Migration discovers the location for ASM and ACFS `data`, `redo`, and `reco` storage volumes from the specified target database, making these target database storage properties optional.

If you need to override the values automatically discovered by Zero Downtime Migration, then you can set the following parameters. For example, `TGT_DATADG=+DATAC3`

For ASM use these parameters

`TGT_DATADG`

`TGT_REDODG`

`TGT_RECODG`

For ACFS use these parameters

`TGT_DATAACFS`

`TGT_REDOACFS`

`TGT_RECOACFS`

TGT_REDOACFS

`TGT_REDOACFS` specifies the location for redo log files ACFS volume (`redo`) on the target database. Use only if required to override the values discovered automatically by Zero Downtime Migration.

Property	Description
Syntax	`TGT_REDOACFS = ` *`redo_location`*
Default value	There is no default value.

Usage Notes

Zero Downtime Migration discovers the location for ASM and ACFS `data`, `redo`, and `reco` storage volumes from the specified target database, making these target database storage properties optional.

If you need to override the values automatically discovered by Zero Downtime Migration, then you can set the following parameters. For example, `TGT_DATADG=+DATAC3`

For ASM use these parameters

`TGT_DATADG`

`TGT_REDODG`

`TGT_RECODG`

For ACFS use these parameters

`TGT_DATAACFS`

`TGT_REDOACFS`

`TGT_RECOACFS`

TGT_REDODG

`TGT_REDODG` specifies the location for redo log files ASM disk group (`redo`) on the target database. Use only if required to override the values discovered automatically by Zero Downtime Migration.

Property	Description
Syntax	`TGT_REDODG = `*`redo_location`*
Default value	There is no default value.

Usage Notes

Zero Downtime Migration discovers the location for ASM and ACFS `data`, `redo`, and `reco` storage volumes from the specified target database, making these target database storage properties optional.

If you need to override the values automatically discovered by Zero Downtime Migration, then you can set the following parameters. For example, `TGT_DATADG=+DATAC3`

For ASM use these parameters

`TGT_DATADG`

`TGT_REDODG`

`TGT_RECODG`

For ACFS use these parameters

`TGT_DATAACFS`

`TGT_REDOACFS`

`TGT_RECOACFS`

TGT_RETAIN_DB_UNIQUE_NAME

`TGT_RETAIN_DB_UNIQUE_NAME` specifies whether to ship redo logs from Oracle Cloud to the on-premises standby, observe the environment for some time, and remove the fallback later.

When `TGT_RETAIN_DB_UNIQUE_NAME=TRUE` then the workflow phase `ZDM_RETAIN_DBUNIQUENAME_TGT` is present as the final phase. You must pause the migration job at a prior phase and resume the job when the fallback has to be removed.

Property	Description
Syntax	`TGT_RETAIN_DB_UNIQUE_NAME = {TRUE | FALSE}`
Default value	`FALSE`
Range of values	`TRUE` - enables this feature `FALSE` - disables this feature

TGT_RMAN_CHANNELS

`TGT_RMAN_CHANNELS` specifies the number of RMAN channels to be allocated at the target database server for performing RMAN restore.

Property	Description
Syntax	`TGT_RMAN_CHANNELS =` *`number_of_channels`*
Default value	`10`

TGT_SKIP_DATAPATCH

`TGT_SKIP_DATAPATCH` specifies whether or not Zero Downtime Migration runs the datapatch utility on the target database as part of the post-migration tasks.

Property	Description
Syntax	`TGT_SKIP_DATAPATCH = {TRUE | FALSE}`
Default value	`FALSE`
Range of values	`TRUE` - do not allow Zero Downtime Migration to run datapatch `FALSE` - allow Zero Downtime Migration to run datapatch

Usage Notes

If the target database environment is at a higher patch level than the source database (for example, if the source database is at Jan 2020 PSU/BP and the target database is at April 2020 PSU/BP), then set the `TGT_SKIP_DATAPATCH` parameter to `FALSE` to allow Zero Downtime Migration to run the datapatch utility on the target database as part of the post-migration tasks.

Otherwise, set the parameter to `TRUE`, and if the target database environment is at a higher patch level than the source database, you will need to run the datapatch utility manually after the migration.

TGT_SSH_RETRY_TIMEOUT

`TGT_SSH_RETRY_TIMEOUT` specifies the number of minutes during which retries are attempted after SSH connection failures. Retries stop when the timeout value has elapsed.

Property	Description
Syntax	`TGT_SSH_RETRY_TIMEOUT =` *`number_of_minutes`*
Default value	There is no default value.

TGT_SSH_TUNNEL_PORT

`TGT_SSH_TUNNEL_PORT` specifies the forwarding port on the source database server where the SSH tunnel to the target database server for SQL*Net connection is set up.

Property	Description
Syntax	`TGT_SSH_TUNNEL_PORT =` *`ssh_tunnel_port_number`*
Default value	There is no default value.

TGT_ZDLRA_WALLET_LOC

`TGT_ZDLRA_WALLET_LOC` specifies the path of the Zero Data Loss Recovery Appliance wallet on the target database server.

Property	Description
Syntax	`TGT_ZDLRA_WALLET_LOC =` *`target_zdlra_wallet_location`* The expected format for the location is `/u02/app/oracle/product/12.1.0/dbhome_3/dbs/zdlra`
Default value	There is no default value.

Usage Notes

When using Zero Data Loss Recovery Appliance as the migration backup medium, you must set the following parameters.

`SRC_ZDLRA_WALLET_LOC`

`TGT_ZDLRA_WALLET_LOC`

`ZDLRA_CRED_ALIAS`

ZDLRA_CRED_ALIAS

`ZDLRA_CRED_ALIAS` specifies the Zero Data Loss Recovery Appliance wallet credential alias.

Property	Description
Syntax	`ZDLRA_CRED_ALIAS =` *`zdlra_wallet_alias`* The expected format for the alias is *`zdlra scan`*`:`*`listener port`*`/zdlra9:dedicated`
Default value	There is no default value.

Usage Notes

When using Zero Data Loss Recovery Appliance as the migration backup medium, you must set the following parameters.

`SRC_ZDLRA_WALLET_LOC`

`TGT_ZDLRA_WALLET_LOC`

`ZDLRA_CRED_ALIAS`

ZDM_BACKUP_DIFFERENTIAL_SRC_MONITORING_INTERVAL

`ZDM_BACKUP_DIFFERENTIAL_SRC_MONITORING_INTERVAL` specifies the time interval, in minutes, at which to monitor and report the progress of the `ZDM_BACKUP_DIFFERENTIAL_SRC` migration job phase.

Property	Description
Syntax	`ZDM_BACKUP_DIFFERENTIAL_SRC_MONITORING_INTERVAL =` *`minutes`*
Default value	`10`

Usage Notes

The migration job phase monitoring interval parameters, listed below, monitor and report the backup and restore operations progress at the set time interval, specified in minutes. Note that the migration job phase for which the monitoring interval applies is prefixed to `_MONITORING_INTERVAL` in each parameter listed above.

- ZDM_BACKUP_FULL_SRC_MONITORING_INTERVAL
- ZDM_BACKUP_INCREMENTAL_SRC_MONITORING_INTERVAL
- ZDM_BACKUP_DIFFERENTIAL_SRC_MONITORING_INTERVAL
- ZDM_CLONE_TGT_MONITORING_INTERVAL
- ZDM_OSS_RESTORE_TGT_MONITORING_INTERVAL
- ZDM_OSS_RECOVER_TGT_MONITORING_INTERVAL

To disable a monitoring interval parameter, set it to `0` (zero).

ZDM_BACKUP_FULL_SRC_MONITORING_INTERVAL

`ZDM_BACKUP_FULL_SRC_MONITORING_INTERVAL` specifies the time interval, in minutes, at which to monitor and report the progress of the `ZDM_BACKUP_FULL_SRC` migration job phase.

Property	Description
Syntax	`ZDM_BACKUP_FULL_SRC_MONITORING_INTERVAL = `*`minutes`*
Default value	`10`

Usage Notes

The migration job phase monitoring interval parameters, listed below, monitor and report the backup and restore operations progress at the set time interval, specified in minutes. Note that the migration job phase for which the monitoring interval applies is prefixed to `_MONITORING_INTERVAL` in each parameter listed above.

- ZDM_BACKUP_FULL_SRC_MONITORING_INTERVAL
- ZDM_BACKUP_INCREMENTAL_SRC_MONITORING_INTERVAL
- ZDM_BACKUP_DIFFERENTIAL_SRC_MONITORING_INTERVAL
- ZDM_CLONE_TGT_MONITORING_INTERVAL
- ZDM_OSS_RESTORE_TGT_MONITORING_INTERVAL
- ZDM_OSS_RECOVER_TGT_MONITORING_INTERVAL

To disable a monitoring interval parameter, set it to `0` (zero).

ZDM_BACKUP_INCREMENTAL_SRC_MONITORING_INTERVAL

`ZDM_BACKUP_INCREMENTAL_SRC_MONITORING_INTERVAL` specifies the time interval, in minutes, at which to monitor and report the progress of the `ZDM_BACKUP_INCREMENTAL_SRC` migration job phase.

Property	Description
Syntax	`ZDM_BACKUP_INCREMENTAL_SRC_MONITORING_INTERVAL = `*`minutes`*
Default value	`10`

Usage Notes

The migration job phase monitoring interval parameters, listed below, monitor and report the backup and restore operations progress at the set time interval, specified in minutes. Note that the migration job phase for which the monitoring interval applies is prefixed to `_MONITORING_INTERVAL` in each parameter listed above.

- ZDM_BACKUP_FULL_SRC_MONITORING_INTERVAL

- ZDM_BACKUP_INCREMENTAL_SRC_MONITORING_INTERVAL
- ZDM_BACKUP_DIFFERENTIAL_SRC_MONITORING_INTERVAL
- ZDM_CLONE_TGT_MONITORING_INTERVAL
- ZDM_OSS_RESTORE_TGT_MONITORING_INTERVAL
- ZDM_OSS_RECOVER_TGT_MONITORING_INTERVAL

To disable a monitoring interval parameter, set it to `0` (zero).

ZDM_BACKUP_RETENTION_WINDOW

`ZDM_BACKUP_RETENTION_WINDOW` specifies the number of days after which backups created by Zero Downtime Migration become obsolete.

Property	Description
Syntax	`ZDM_BACKUP_RETENTION_WINDOW = `*`days`*
Default value	`60`

ZDM_CLONE_TGT_MONITORING_INTERVAL

`ZDM_CLONE_TGT_MONITORING_INTERVAL` specifies the time interval, in minutes, at which to monitor and report the progress of the `ZDM_CLONE_TGT` migration job phase.

Property	Description
Syntax	`ZDM_CLONE_TGT_MONITORING_INTERVAL = `*`minutes`*
Default value	`10`

Usage Notes

The migration job phase monitoring interval parameters, listed below, monitor and report the backup and restore operations progress at the set time interval, specified in minutes. Note that the migration job phase for which the monitoring interval applies is prefixed to `_MONITORING_INTERVAL` in each parameter listed above.

- ZDM_BACKUP_FULL_SRC_MONITORING_INTERVAL
- ZDM_BACKUP_INCREMENTAL_SRC_MONITORING_INTERVAL
- ZDM_BACKUP_DIFFERENTIAL_SRC_MONITORING_INTERVAL
- ZDM_CLONE_TGT_MONITORING_INTERVAL
- ZDM_OSS_RESTORE_TGT_MONITORING_INTERVAL
- ZDM_OSS_RECOVER_TGT_MONITORING_INTERVAL

To disable a monitoring interval parameter, set it to `0` (zero).

ZDM_CURL_LOCATION

`ZDM_CURL_LOCATION` specifies a custom location for the CURL binary on the source.

Property	Description
Syntax	`ZDM_CURL_LOCATION = curl_location`
Default value	`/usr/bin/curl`

ZDM_LOG_OSS_PAR_URL

`ZDM_LOG_OSS_PAR_URL` specifies the pre-authenticated URL to use when uploading logs to Object Storage Service. The logs capture the current migration job phase and the execution status of the phase.

Property	Description
Syntax	`ZDM_LOG_OSS_PAR_URL = url`
Default value	There is no default value. By default this parameter is disabled.

ZDM_OPC_RETRY_COUNT

`ZDM_OPC_RETRY_COUNT` specifies the number of retry attempts tat will be made after an initial Object Store connection failure.

Property	Description
Syntax	`ZDM_OPC_RETRY_COUNT = number`
Default value	`0` (zero) The default behavior is to attempt no retries.

ZDM_OPC_RETRY_WAIT_TIME

`ZDM_OPC_RETRY_WAIT_TIME` specifies the number of seconds to wait after an Object Store connection failure before attempting to retry the connection.

Property	Description
Syntax	`ZDM_OPC_RETRY_WAIT_TIME = seconds`
Default value	`529` (seconds)

ZDM_OSS_RECOVER_TGT_MONITORING_INTERVAL

`ZDM_OSS_RECOVER_TGT_MONITORING_INTERVAL` specifies the time interval, in minutes, at which to monitor and report the progress of the `ZDM_OSS_RECOVER_TGT` migration job phase.

Property	Description
Syntax	`ZDM_OSS_RECOVER_TGT_MONITORING_INTERVAL = minutes`

Property	Description
Default value	`10`

Usage Notes

The migration job phase monitoring interval parameters, listed below, monitor and report the backup and restore operations progress at the set time interval, specified in minutes. Note that the migration job phase for which the monitoring interval applies is prefixed to `_MONITORING_INTERVAL` in each parameter listed above.

- ZDM_BACKUP_FULL_SRC_MONITORING_INTERVAL
- ZDM_BACKUP_INCREMENTAL_SRC_MONITORING_INTERVAL
- ZDM_BACKUP_DIFFERENTIAL_SRC_MONITORING_INTERVAL
- ZDM_CLONE_TGT_MONITORING_INTERVAL
- ZDM_OSS_RESTORE_TGT_MONITORING_INTERVAL
- ZDM_OSS_RECOVER_TGT_MONITORING_INTERVAL

To disable a monitoring interval parameter, set it to `0` (zero).

ZDM_OSS_RESTORE_TGT_MONITORING_INTERVAL

`ZDM_OSS_RESTORE_TGT_MONITORING_INTERVAL` specifies the time interval, in minutes, at which to monitor and report the progress of the `ZDM_OSS_RESTORE_TGT` migration job phase.

Property	Description
Syntax	`ZDM_OSS_RESTORE_TGT_MONITORING_INTERVAL = *minutes*`
Default value	`10`

Usage Notes

The migration job phase monitoring interval parameters, listed below, monitor and report the backup and restore operations progress at the set time interval, specified in minutes. Note that the migration job phase for which the monitoring interval applies is prefixed to `_MONITORING_INTERVAL` in each parameter listed above.

- ZDM_BACKUP_FULL_SRC_MONITORING_INTERVAL
- ZDM_BACKUP_INCREMENTAL_SRC_MONITORING_INTERVAL
- ZDM_BACKUP_DIFFERENTIAL_SRC_MONITORING_INTERVAL
- ZDM_CLONE_TGT_MONITORING_INTERVAL
- ZDM_OSS_RESTORE_TGT_MONITORING_INTERVAL
- ZDM_OSS_RECOVER_TGT_MONITORING_INTERVAL

To disable a monitoring interval parameter, set it to `0` (zero).

ZDM_RMAN_COMPRESSION_ALGORITHM

`ZDM_RMAN_COMPRESSION_ALGORITHM` specifies which RMAN compression algorithm to use for backups.

Property	Description
Syntax	`ZDM_RMAN_COMPRESSION_ALGORITHM ={BASIC | LOW | MEDIUM | HIGH}`
Default value	`MEDIUM`
Range of values	`BASIC` - Does not require the Advanced Compression Option (ACO). `LOW` - Least impact on backup throughput and suited for environments where CPU resources are the limiting factor. `MEDIUM` - Recommended for most environments. Good combination of compression ratios and speed `HIGH` - Best suited for backups over slower networks where the limiting factor is network speed

ZDM_SHARD_ID

`ZDM_SHARD_ID` is used to ensure that a migration job is running or is being resumed in the correct intended pod.

In a scenario where Zero Downtime Migration is scaled out to service multiple simultaneous migrations, each ZDM server has its own metadata store, which means that the same migration job ID values could be repeated across multiple ZDM servers.

To avoid job ID conflicts which could cause the incorrect job to be resumed on the incorrect ZDM server, the `ZDM_SHARD_ID` for each job will be configured contain the ZDM host name to which the migration job is being sent. This value will be seeded by E2E during RSP creation.

The value of `ZDM_SHARD_ID` and the other RSP tokens will be used by ZDM to confirm that the job metadata matches the RSP file values for the source and target database properties to ensure the correct job ID and ZDM server are resumed.

If the value from `ZDM_SHARD_ID` in the response file is set, and it does not match the current host name, the following exception is thrown:

```
PRGZ-#### : Specified shard ID zdm_host_name_a does not match with current
ZDM host zdm_host_name_b.
```

in which den01gl is the value read from the response file, and den01glt is the current pod's host name

Property	Description
Syntax	`ZDM_SHARD_ID =` *`zdm_host_name`*

Property	Description
Default value	No default value By default, ZDM will not consider the host name before starting or resuming a job.
Range of values	`ZDM_SHARD_ID` accepts the host name of the pod in which the current job will be run or resumed. It accepts host name or fully qualified domain name. The `ZDM_SHARD_ID` verification is case insensitive

ZDM_SKIP_DG_CONFIG_CLEANUP

`ZDM_SKIP_DG_CONFIG_CLEANUP` indicates whether Zero Downtime Migration should clean up the Oracle Data Guard configuration from the source and target databases at the end of the migration when using online physical migration.

Property	Description
Syntax	`ZDM_SKIP_DG_CONFIG_CLEANUP ={TRUE | FALSE}`
Default value	`FALSE` By default, ZDM will deconfigure Data Guard parameters configured for migration.
Range of values	`TRUE` - Do not clean up the Oracle Data Guard configuration. `FALSE` - Clean up the Oracle Data Guard configuration.

ZDM_SRC_TNS_ADMIN

`ZDM_SRC_TNS_ADMIN` specifies the custom location for `TNS_ADMIN` on the source database server when there is no Oracle Grid Infrastructure. If a Grid Infrastructure exists, then the `TNS_ADMIN` property must be set in the CRS resource attribute environment of the database resource.

Property	Description
Syntax	`ZDM_SRC_TNS_ADMIN =` *`tns_admin_location`*
Default value	There is no default value.

ZDM_USE_EXISTING_UNDO_SIZE

`ZDM_USE_EXISTING_UNDO_SIZE` specifies whether Zero Downtime Migration should use the existing undo tablespace size when creating a new undo tablespace, if required.

Property	Description
Syntax	`ZDM_USE_EXISTING_UNDO_SIZE ={TRUE | FALSE}`
Default value	`TRUE` By default, Zero Downtime Migration uses the largest size of the existing undo tablespaces.
Range of values	`TRUE` - Use the existing undo tablespace size. `FALSE` - Do not use the existing undo tablespace size.

G

Zero Downtime Migration Logical Migration Response File Parameters Reference

The following topics describe the Zero Downtime Migration response file parameters supported for logical migrations.

DATA_TRANSFER_MEDIUM

In a logical migration, the optional `DATA_TRANSFER_MEDIUM` parameter specifies how data will be transferred from the source database system to the target database system for initial load.

Property	Description
Syntax	`DATA_TRANSFER_MEDIUM = {OSS | NFS | DBLINK | COPY}`
Default value	`OSS`
Range of values	`OSS` - Object Storage Service `NFS` - Network File System `DBLINK` - Direct transfer of data over a database link `COPY` - secure copy (for user-managend targets only)

DATAPUMPSETTINGS_CREATEAUTHTOKEN

`DATAPUMPSETTINGS_CREATEAUTHTOKEN` is one of the optional settings for logical migrations using Data Pump.

If you are not using a network database link for Data Pump Import, an OCI Auth Token will be created for the specified OCI user to import Data Pump dump files from the Object Storage into an Autonomous Database. To reuse an existing Auth Token, set this property to `FALSE`.

Property	Description
Syntax	`DATAPUMPSETTINGS_CREATEAUTHTOKEN ={TRUE | FALSE}`
Default value	`FALSE`
Range of values	`TRUE` - Creates an OCI Auth Token for the specified OCI user `FALSE` - Does not create an OCI Auth Token.

DATAPUMPSETTINGS_DATABASELINKDETAILS_NAME

`DATAPUMPSETTINGS_DATABASELINKDETAILS_NAME` specifies the name of the database link from the OCI database to the on-premise database. ZDM creates the database link if the link does not already exist.

The `DATAPUMPSETTINGS_DATABASELINKDETAILS` parameters are optional details for creating a network database link from OCI database to the on-premise database. It is one of the optional settings for logical migrations using Data Pump.

Property	Description
Syntax	`DATAPUMPSETTINGS_DATABASELINKDETAILS_NAME = ` *`db_link_name`*
Default value	There is no default value
Range of values	A string value is expected

DATAPUMPSETTINGS_DATABASELINKDETAILS_WALLETBUCKET_BUCKETNAME

`DATAPUMPSETTINGS_DATABASELINKDETAILS_WALLETBUCKET_BUCKETNAME` specifies the OCI Object Storage bucket.

The `DATAPUMPSETTINGS_DATABASELINKDETAILS_WALLETBUCKET` parameters, in the case of Autonomous Database, specifies the OCI Object Storage details used to store the wallet containing the certificate for on-premise database to create a database link from the Autonomous Database to the on-premise database using TLS.

Not required for a TCP connection from Autonomous Database to the on-premise database.

Property	Description
Syntax	`DATAPUMPSETTINGS_DATABASELINKDETAILS_WALLETBUCKET_BUCKETNAME = ` *`Object Storage bucket name`*
Default value	There is no default value
Range of values	A string value is expected

DATAPUMPSETTINGS_DATABASELINKDETAILS_WALLETBUCKET_NAMESPACENAME

`DATAPUMPSETTINGS_DATABASELINKDETAILS_WALLETBUCKET_NAMESPACENAME` specifies the Object Storage namespace.

The `DATAPUMPSETTINGS_DATABASELINKDETAILS_WALLETBUCKET` parameters, in the case of Autonomous Database, specifies the OCI Object Storage bucket used to store the wallet containing the certificate for on-premise database to create a database link from the Autonomous Database to the on-premise database using TLS.

Not required for a TCP connection from Autonomous Database to the on-premise database.

Property	Description
Syntax	`DATAPUMPSETTINGS_DATABASELINKDETAILS_WALLETBUCKET_NAMESPACENAME = `*`Object Storage bucket namespace`*
Default value	There is no default value
Range of values	A string value is expected

DATAPUMPSETTINGS_DATABUCKET_BUCKETNAME

`DATAPUMPSETTINGS_DATABUCKET_BUCKETNAME` is one of the optional settings for logical migrations using Data Pump.

In lieu of a network database link, this OCI Object Storage bucket will be used to store Data Pump dump files for migrating to an Autonomous Database. Use with `DATAPUMPSETTINGS_DATABUCKET_NAMESPACENAME`

Property	Description
Syntax	`DATAPUMPSETTINGS_DATABUCKET_BUCKETNAME = `*`Name of the Object Storage bucket`*
Default value	There is no default value
Range of values	Enter the storeage bucket name

DATAPUMPSETTINGS_DATABUCKET_NAMESPACENAME

`DATAPUMPSETTINGS_DATABUCKET_NAMESPACENAME` is one of the optional settings for logical migrations using Data Pump.

In lieu of a network database link, this OCI Object Storage bucket will be used to store Data Pump dump files for migrating to an Autonomous Database. Use with `DATAPUMPSETTINGS_DATABUCKET_BUCKETNAME`

Property	Description
Syntax	`DATAPUMPSETTINGS_DATABUCKET_NAMESPACENAME = `*`Object Storage namespace`*
Default value	There is no default value
Range of values	Enter the storeage bucket namespace

DATAPUMPSETTINGS_DATAPUMPPARAMETERS_TABLEEXISTSACTION

`DATAPUMPSETTINGS_DATAPUMPPARAMETERS_TABLEEXISTSACTION` specifies the action to be performed when data is loaded into a preexisting table.

`DATAPUMPSETTINGS_DATAPUMPPARAMETERS` are optional parameters for Data Pump Export and Import. See SET_PARAMETER Procedures for more information.

Property	Description
Syntax	`DATAPUMPSETTINGS_DATAPUMPPARAMETERS_TABLEEXISTSACTION = [SKIP | TRUNCATE | REPLACE | APPEND]`
Default value	`SKIP`
Range of values	`SKIP` - the preexisting table is left unchanged. `TRUNCATE` - rows are removed from a preexisting table before inserting rows from the Import. Note that if `TRUNCATE` is specified on tables referenced by foreign key constraints, the `TRUNCATE` will be modified into a `REPLACE`. `REPLACE` - preexisting tables are replaced with new definitions. Before creating the new table, the old table is dropped. `APPEND` - new rows are added to the existing rows in the table See `TABLE_EXISTS_ACTION` entry in Table 48-25 "Valid Options for the name Parameter in the SET_PARAMETER Procedure" in SET_PARAMETER Procedures

DATAPUMPSETTINGS_DATAPUMPPARAMETERS_USERMETADATA

`DATAPUMPSETTINGS_DATAPUMPPARAMETERS_USERMETADATA` is used for EXPORT and Network IMPORT. If set to a nonzero value for schema-mode operations, it specifies that the metadata to re-create the user's schemas should also be part of the operation.

`DATAPUMPSETTINGS_DATAPUMPPARAMETERS` are optional parameters for Data Pump Export and Import. See SET_PARAMETER Procedures for more information.

Property	Description
Syntax	`DATAPUMPSETTINGS_DATAPUMPPARAMETERS_USERMETADATA = `*`integer`*
Default value	There is no default value
Valid values	An integer value is expected

DATAPUMPSETTINGS_DATAPUMPPARAMETERS_IMPORTPARALLELISMDEGREE

`DATAPUMPSETTINGS_DATAPUMPPARAMETERS_IMPORTPARALLELISMDEGREE` specifies the maximum number of worker processes that can be used for a Data Pump Import job. For migration to an Autonomous Database target, ZDM automatically queries its CPU core count and sets this parameter.

`DATAPUMPSETTINGS_DATAPUMPPARAMETERS` are optional parameters for Data Pump Export and Import. See SET_PARAMETER Procedures for more information.

Property	Description
Syntax	`DATAPUMPSETTINGS_DATAPUMPPARAMETERS_IMPORTPARALLELISMDEGREE = integer`
Default value	There is no default value
Valid values	An integer value is expected

DATAPUMPSETTINGS_DATAPUMPPARAMETERS_EXPORTPARALLELISMDEGREE

`DATAPUMPSETTINGS_DATAPUMPPARAMETERS_EXPORTPARALLELISMDEGREE` specifies the maximum number of worker processes that can be used for a Data Pump Import job. For migration to an Autonomous Database target, ZDM automatically queries its CPU core count and sets this parameter.

`DATAPUMPSETTINGS_DATAPUMPPARAMETERS` are optional parameters for Data Pump Export and Import. See SET_PARAMETER Procedures for more information.

Property	Description
Syntax	`DATAPUMPSETTINGS_DATAPUMPPARAMETERS_EXPORTPARALLELISMDEGREE = integer`
Default value	There is no default value
Valid values	An integer value is expected

DATAPUMPSETTINGS_DATAPUMPPARAMETERS_EXCLUDETYPELIST

`DATAPUMPSETTINGS_DATAPUMPPARAMETERS_EXCLUDETYPELIST` specifies a comma separated list of object types to exclude.

`DATAPUMPSETTINGS_DATAPUMPPARAMETERS` are optional parameters for Data Pump Export and Import. See SET_PARAMETER Procedures for more information.

Property	Description
Syntax	`DATAPUMPSETTINGS_DATAPUMPPARAMETERS_EXCLUDETYPELIST = object_type_list`
Default value	There is no default
Valid values	A comma separated list of object types

Example

```
DATAPUMPSETTINGS_DATAPUMPPARAMETERS_EXCLUDETYPELIST=cluster,dblink,comment
```

DATAPUMPSETTINGS_DATAPUMPPARAMETERS_SKIPCURRENT

`DATAPUMPSETTINGS_DATAPUMPPARAMETERS_SKIPCURRENT`, when enabled, causes actions that were 'in progress' on a previous execution of the job to be skipped when the job restarts. The skip is only honored for Import jobs.

This mechanism allows you to skip actions that trigger fatal bugs and cause the premature termination of a job. Multiple actions can be skipped on a restart. The log file will identify which actions are skipped. If a domain index was being processed, all pieces of the domain index are skipped, even if the error only occurred in a sub-component of the domain index.

`DATAPUMPSETTINGS_DATAPUMPPARAMETERS` are optional parameters for Data Pump Export and Import. See SET_PARAMETER Procedures for more information.

Property	Description
Syntax	`DATAPUMPSETTINGS_DATAPUMPPARAMETERS_SKIPCURRENT = [TRUE | FALSE]`
Default value	`FALSE`
Valid values	`TRUE` enables this parameter `FALSE` disables this parameter

DATAPUMPSETTINGS_DATAPUMPPARAMETERS_NOCLUSTER

`DATAPUMPSETTINGS_DATAPUMPPARAMETERS_NOCLUSTER`, when enabled, all Data Pump workers are started on the current instance; otherwise, workers are started on instances usable by the job.

`DATAPUMPSETTINGS_DATAPUMPPARAMETERS` are optional parameters for Data Pump Export and Import. See SET_PARAMETER Procedures for more information.

Property	Description
Syntax	`DATAPUMPSETTINGS_DATAPUMPPARAMETERS_NOCLUSTER = [TRUE | FALSE]`
Default value	`FALSE`
Valid values	`TRUE` enables this parameter `FALSE` disables this parameter

DATAPUMPSETTINGS_DATAPUMPPARAMETERS_RETAININDEX

`DATAPUMPSETTINGS_DATAPUMPPARAMETERS_RETAININDEX`, when enabled, retains the index.

`DATAPUMPSETTINGS_DATAPUMPPARAMETERS` are optional parameters for Data Pump Export and Import. See SET_PARAMETER Procedures for more information.

Property	Description
Syntax	`DATAPUMPSETTINGS_DATAPUMPPARAMETERS_RETAININDEX = [TRUE	FALSE]`
Default value	`FALSE`
Valid values	`TRUE` enables this parameter `FALSE` disables this parameter

DATAPUMPSETTINGS_DELETEDUMPSINOSS

`DATAPUMPSETTINGS_DELETEDUMPSINOSS` is used to indicate whether dump files which are uploaded to Object Storage as part of the migration are retained.

Property	Description
Syntax	`DATAPUMPSETTINGS_DELETEDUMPSINOSS = {TRUE|FALSE}`
Default value	`TRUE`
Range of values	`TRUE` = delete dumps `FALSE` = retain dumps

DATAPUMPSETTINGS_EXPORTDIRECTORYOBJECT_NAME

In lieu of a network database link, this directory on the server file system of an on-premises database is used to store Data Pump Export dump files. `DATAPUMPSETTINGS_EXPORTDIRECTORYOBJECT_NAME` specifies the object name. ZDM creates this object if it does not already exist

Property	Description
Syntax	`DATAPUMPSETTINGS_EXPORTDIRECTORYOBJECT_NAME =` *`Name`*
Default value	There is no default value
Range of values	Name of directory object in the database.

DATAPUMPSETTINGS_EXPORTDIRECTORYOBJECT_PATH

In lieu of a network database link, this directory on the server file system of an on-premises database is used to store Data Pump Export dump files. `DATAPUMPSETTINGS_EXPORTDIRECTORYOBJECT_PATH` specifies the object path. ZDM creates this object if it does not already exist.

Note:

For Oracle Database 19c and later releases, the `UTL_FILE_DIR` initialization parameter is desupported, which means symbolic links for Data Pump directories are not supported.
If you attempt to use an affected feature configured with symbolic links, then you encounter ORA-29283: invalid file operation: path traverses a symlink. Oracle recommends that you instead use directory objects in place of symbolic links.

See How Does Oracle Data Pump Move Data? for more information.

Property	Description
Syntax	`DATAPUMPSETTINGS_EXPORTDIRECTORYOBJECT_PATH = *Path*`
Default value	There is no default value
Range of values	Absolute path of directory on database server

DATAPUMPSETTINGS_FIXINVALIDOBJECTS

`DATAPUMPSETTINGS_FIXINVALIDOBJECTS` specifies whether invalid objects are recompiled in the database as part of the ZDM Data Pump migration procedure.

Migration can result in invalid schema objects. Typically, invalid objects fix themselves as they are accessed or run. However, Oracle recommends that invalid objects are recompiled in the database as part of the ZDM migration procedure, so that issues with invalid objects, and any required dependencies are resolved before users encounter these invalid objects.

Property	Description
Syntax	`DATAPUMPSETTINGS_FIXINVALIDOBJECTS = {TRUE | FALSE}`
Default value	`FALSE`
Range of values	`TRUE` = fix invalid objects `FALSE` = do not fix invalid objects

DATAPUMPSETTINGS_IMPORTDIRECTORYOBJECT_NAME

`DATAPUMPSETTINGS_IMPORTDIRECTORYOBJECT_NAME` specifies the name of an import directory object.

In lieu of a network database link, this directory on the server file system of OCI database will be used to store Data Pump dump files. ZDM will create this object if it does not already exist. In case of Autonomous Database, `DATA_PUMP_DIR` object will already exist

.

Property	Description
Syntax	`DATAPUMPSETTINGS_IMPORTDIRECTORYOBJECT_NAME = ` *`Name of the directory object`*
Default value	There is no default value
Range of values	Name of directory object in database

DATAPUMPSETTINGS_IMPORTDIRECTORYOBJECT_PATH

`DATAPUMPSETTINGS_IMPORTDIRECTORYOBJECT_PATH` specifies the path of an import directory object.

In lieu of a network database link, this directory on the server file system of OCI database will be used to store Data Pump dump files. ZDM will create this object if it does not already exist. In case of Autonomous Database, `DATA_PUMP_DIR` object will already exist.

Note:

For Oracle Database 19c and later releases, the `UTL_FILE_DIR` initialization parameter is desupported, which means symbolic links for Data Pump directories are not supported.
If you attempt to use an affected feature configured with symbolic links, then you encounter ORA-29283: invalid file operation: path traverses a symlink. Oracle recommends that you instead use directory objects in place of symbolic links.

See How Does Oracle Data Pump Move Data? for more information.

Property	Description
Syntax	`DATAPUMPSETTINGS_IMPORTDIRECTORYOBJECT_PATH = ` *`Path of the directory object`*
Default value	There is no default value
Range of values	Absolute path of directory on database server

DATAPUMPSETTINGS_JOBMODE

`DATAPUMPSETTINGS_JOBMODE` specifies the Data Pump export mode.

.

Property	Description
Syntax	`DATAPUMPSETTINGS_JOBMODE =` *`jobModeValue`*
Default value	`SCHEMA`
Range of values	`FULL` performs a full database export. `SCHEMA` (default) lets you specify a set of schemas to export. `TABLE` lets you specify a set of tables to export. In this mode, Zero Downtime Migration precreates the target schema before Data Pump import. `TABLESPACE` lets you specify a set of tablespaces to export. In this mode, Zero Downtime Migration precreates the target schema before Data Pump import. `TRANSPORTABLE` is not supported by Zero Downtime Migration. See Oracle Data Pump Export Modes for more information.

DATAPUMPSETTINGS_METADATAFILTERS-*LIST_ELEMENT_NUMBER*

`DATAPUMPSETTINGS_METADATAFILTERS-`*`LIST_ELEMENT_NUMBER`* defines the name, the object type, and the value of the filter for the Data Pump `METADATA_FILTER` property.

To add multiple filters, increment the integer appended to the parameter name, as shown in the examples below.

```
DATAPUMPSETTINGS_METADATAFILTERS-1=name:nameValue1st,
objectType:objectTypeValue1st, value:valueValue1st
DATAPUMPSETTINGS_METADATAFILTERS-2=name:nameValue2nd,
objectType:objectTypeValue2nd, value:valueValue2nd
```

To exclude select `SCHEMA` Objects for `FULL` mode:

```
DATAPUMPSETTINGS_METADATAFILTERS-1=name:NAME_EXPR,value:'NOT IN('
'SYSMAN' ')',objectType:SCHEMA
DATAPUMPSETTINGS_METADATAFILTERS-2=name:NAME_EXPR,value:'NOT IN(' 'SH'
')',objectType:SCHEMA
```

Note that the `SCHEMA` name `SYSMAN` is surrounded by two single quotes and not a double quote.

Property	Description
Syntax	`DATAPUMPSETTINGS_METADATAFILTERS-`*`LIST_ELEMENT_NUMBER`* `= name:`*`nameValue`*`, objectType:`*`objectTypeValue`*`, value:`*`valueValue`*
Default value	There is no default value
Range of values	An entry specifying the name, type, and value is expected, as shown in the examples above.

DATAPUMPSETTINGS_METADATAREMAPS-*LIST_ELEMENT_NUMBER*

`DATAPUMPSETTINGS_METADATAREMAPS-`*`LIST_ELEMENT_NUMBER`* defines remapping to be applied to objects as they are processed.

To add multiple remappings, increment the integer appended to the parameter name, as shown in the examples below.

```
DATAPUMPSETTINGS_METADATAREMAPS-1=type:typeValue1st,
oldValue:oldValueValue1st, newValue:newValueValue1st
DATAPUMPSETTINGS_METADATAREMAPS-2=type:typeValue2nd,
oldValue:oldValueValue2nd, newValue:newValueValue2nd
```

See METADATA_REMAP Procedure for more information.

Property	Description
Syntax	`DATAPUMPSETTINGS_METADATAREMAPS-`*`LIST_ELEMENT_NUMBER`* `= newValue:`*`newValueValue`*`, oldValue:`*`oldValueValue`*`, type:`*`typeValue`*
Default value	There is no default value
Range of values	An entry specifying the new value, old value and type is expected, as shown in the examples above.

DATAPUMPSETTINGS_METADATATRANSFORMS-*LIST_ELEMENT_NUMBER*

`DATAPUMPSETTINGS_METADATATRANSFORMS-`*`LIST_ELEMENT_NUMBER`* defines the name, the object type, and the value for the Data Pump `METADATA_TRANSFORM` property.

To add multiple filters, increment the integer appended to the parameter name, as shown in the examples below.

```
DATAPUMPSETTINGS_METADATATRANSFORMS-1=name:nameValue1st,
objectType:objectTypeValue1st, value:valueValue1st
DATAPUMPSETTINGS_METADATATRANSFORMS-2=name:nameValue2nd,
objectType:objectTypeValue2nd, value:valueValue2nd
```

Property	Description
Syntax	`DATAPUMPSETTINGS_METADATATRANSFORMS-`*`LIST_ELEMENT_NUMBER`* `=` `name:`*`nameValue`*`,` `objectType:`*`objectTypeValue`*`,` `value:`*`valueValue`*
Default value	There is no default value
Range of values	An entry specifying the name, type, and value is expected, as shown in the examples above.

DATAPUMPSETTINGS_MONITORINTERVALMINUTES

`DATAPUMPSETTINGS_MONITORINTERVALMINUTES` specifies the Data Pump monitor interval in minutes. This setting is optional.

.

Property	Description
Syntax	`DATAPUMPSETTINGS_MONITORINTERVALMINUTES =` *`number`*
Default value	10
Range of values	An integer value is expected

DATAPUMPSETTINGS_OMITENCRYPTIONCLAUSE

`DATAPUMPSETTINGS_OMITENCRYPTIONCLAUSE`, when enabled, sets `TRANSFORM=OMIT_ENCRYPTION_CLAUSE`, which directs Data Pump to suppress any encryption clauses associated with objects using encrypted columns.

This parameter is valid for targets on Oracle Database 19c and later releases.

`OMIT_ENCRYPTION_CLAUSE` applies to materialized view, table, and tablespace objects, and enables objects which were using encrypted columns in the source to get created in a target database environment where encryption attributes are not supported.

Property	Description
Syntax	`DATAPUMPSETTINGS_OMITENCRYPTIONCLAUSE = [TRUE | FALSE]`
Default value	`TRUE`
Valid values	`TRUE` enables this parameter `FALSE` disables this parameter

DATAPUMPSETTINGS_SECUREFILELOB

`DATAPUMPSETTINGS_SECUREFILELOB`, when enabled, sets `TRANSFORM=LOB_STORAGE:SECUREFILE`, which directs Data Pump to transform basic LOBs into securefile LOBs during the Data Pump import.

Property	Description
Syntax	`DATAPUMPSETTINGS_SECUREFILELOB = [TRUE | FALSE]`
Default value	`TRUE`
Valid values	`TRUE` enables this parameter `FALSE` disables this parameter

DATAPUMPSETTINGS_SKIPDEFAULTTRANSFORM

`DATAPUMPSETTINGS_SKIPDEFAULTTRANSFORM`, when enabled, skips default transform parameters. Set this property to `TRUE` to avoid all internal ZDM transform defaults.

Property	Description
Syntax	`DATAPUMPSETTINGS_SKIPDEFAULTTRANSFORM = [TRUE | FALSE]`
Default value	`FALSE`
Valid values	`TRUE` enables this parameter `FALSE` disables this parameter

DUMPTRANSFERDETAILS_PARALLELCOUNT

`DUMPTRANSFERDETAILS_PARALLELCOUNT` specifies the maximum number of Export dump files to transfer to Object Storage or remote node in parallel.

The `DUMPTRANSFERDETAILS` parameters specify details to transfer dumps from the source node to the target node. Supported modes of transfer are OSS and secure copy. Upload of dumps to OSS can be either through OCI CLI or Curl.

Property	Description
Syntax	`DUMPTRANSFERDETAILS_PARALLELCOUNT = integer`
Default value	3
Range of values	This parameter accepts integer values

DUMPTRANSFERDETAILS_RETRYCOUNT

`DUMPTRANSFERDETAILS_RETRYCOUNT` specifies the number of times to retry upload or transfer of dump failure.

The `DUMPTRANSFERDETAILS` parameters specify details to transfer dumps from the source node to the target node. Supported modes of transfer are OSS and secure copy. Upload of dumps to OSS can be either through OCI CLI or Curl.

Property	Description
Syntax	`DUMPTRANSFERDETAILS_RETRYCOUNT = integer`
Default value	3
Range of values	This parameter accepts integer values

DUMPTRANSFERDETAILS_RSYNCAVAILABLE

`DUMPTRANSFERDETAILS_RSYNCAVAILABLE` specifies that the transfer dump files are using the Linux rsync utility. Ensure rsync is installed in both source and target servers. Oracle Exadata does not ship with rsync, refer to MOS Doc ID 1556257.1 for details. ZDM defaults to scp command for transfer.

The `DUMPTRANSFERDETAILS` parameters specify details to transfer dumps from the source node to the target node. Supported modes of transfer are OSS and secure copy. Upload of dumps to OSS can be either through OCI CLI or Curl.

Property	Description
Syntax	`DUMPTRANSFERDETAILS_RSYNCAVAILABLE = {TRUE | FALSE`
Default value	`FALSE`
Range of values	`TRUE` - transfer dump files are using the Linux rsync utility `FALSE` - not using rsync

DUMPTRANSFERDETAILS_SOURCE_OCIHOME

`DUMPTRANSFERDETAILS_SOURCE_OCIHOME` specifies the Oracle Cloud OCI-CLI Binary path.

The `DUMPTRANSFERDETAILS` parameters specify details to transfer dumps from the source node to the target node. Supported modes of transfer are OSS and secure copy. Upload of dumps to OSS can be either through OCI CLI or Curl.

The `DUMPTRANSFERDETAILS_SOURCE` parameters specify details to upload the Dump files from source node Data Pump dump location to Oracle Cloud Object Storage.

Property	Description
Syntax	`DUMPTRANSFERDETAILS_SOURCE_OCIHOME = `*`path`*
Default value	No default value
Range of values	A string value is expected

DUMPTRANSFERDETAILS_SOURCE_PARTSIZEMB

`DUMPTRANSFERDETAILS_SOURCE_PARTSIZEMB` specifies the part size in MB for chunked transfer.

The `DUMPTRANSFERDETAILS` parameters specify details to transfer dumps from the source node to the target node. Supported modes of transfer are OSS and secure copy. Upload of dumps to OSS can be either through OCI CLI or Curl.

The `DUMPTRANSFERDETAILS_SOURCE` parameters specify details to upload the Dump files from source node Data Pump dump location to Oracle Cloud Object Storage.

Property	Description
Syntax	`DUMPTRANSFERDETAILS_SOURCE_PARTSIZEMB = `*`integer`*
Default value	`128`
Range of values	An integer value is expected

DUMPTRANSFERDETAILS_SOURCE_TRANSFERNODE_DUMPDIRPATH

`DUMPTRANSFERDETAILS_SOURCE_TRANSFERNODE_DUMPDIRPATH` specifies the absolute path of the directory to copy or upload dump files.

The `DUMPTRANSFERDETAILS` parameters specify details to transfer dumps from the source node to the target node. Supported modes of transfer are OSS and secure copy. Upload of dumps to OSS can be either through OCI CLI or Curl.

The `DUMPTRANSFERDETAILS_SOURCE` parameters specify details to upload the Dump files from source node Data Pump dump location to Oracle Cloud Object Storage.

The `DUMPTRANSFERDETAILS_SOURCE_TRANSFERNODE` parameters are configured when you use a standalone server as transfer server. Transfer dump files using Oracle Cloud OCI-CLI or curl, or copy to or from a node other than the database server. This option can be leveraged for cases where OCI CLI is installed and configured in a specific node per data center and the node has the Data Pump export or import directory path shared with it. This avoids the requirement of installing OCI CLI on the database node.

Property	Description
Syntax	`DUMPTRANSFERDETAILS_SOURCE_TRANSFERNODE_DUMPDIRPATH = `*`path`*
Default value	No default value
Range of values	A string value is expected

DUMPTRANSFERDETAILS_SOURCE_TRANSFERNODE_HOST

`DUMPTRANSFERDETAILS_SOURCE_TRANSFERNODE_HOST` specifies the dump transfer node host name.

The `DUMPTRANSFERDETAILS` parameters specify details to transfer dumps from the source node to the target node. Supported modes of transfer are OSS and secure copy. Upload of dumps to OSS can be either through OCI CLI or Curl.

The `DUMPTRANSFERDETAILS_SOURCE` parameters specify details to upload the Dump files from source node Data Pump dump location to Oracle Cloud Object Storage.

The `DUMPTRANSFERDETAILS_SOURCE_TRANSFERNODE` parameters are configured when you use a standalone server as transfer server. Transfer dump files using Oracle Cloud OCI-CLI or curl, or copy to or from a node other than the database server. This option can be leveraged for cases where OCI CLI is installed and configured in a specific node per data center and the node has the Data Pump export or import directory path shared with it. This avoids the requirement of installing OCI CLI on the database node.

Property	Description
Syntax	`DUMPTRANSFERDETAILS_SOURCE_TRANSFERNODE_HOST = `*`host_name`*
Default value	No default value
Range of values	A string value is expected

DUMPTRANSFERDETAILS_SOURCE_TRANSFERNODE_SUDOPATH

`DUMPTRANSFERDETAILS_SOURCE_TRANSFERNODE_SUDOPATH` specifies the Sudo path on the dump transfer node.

The `DUMPTRANSFERDETAILS` parameters specify details to transfer dumps from the source node to the target node. Supported modes of transfer are OSS and secure copy. Upload of dumps to OSS can be either through OCI CLI or Curl.

The `DUMPTRANSFERDETAILS_SOURCE` parameters specify details to upload the Dump files from source node Data Pump dump location to Oracle Cloud Object Storage.

The `DUMPTRANSFERDETAILS_SOURCE_TRANSFERNODE` parameters are configured when you use a standalone server as transfer server. Transfer dump files using Oracle Cloud OCI-CLI or curl, or copy to or from a node other than the database server. This option can be leveraged for cases where OCI CLI is installed and configured in a specific node per data center and the node has the Data Pump export or import directory path shared with it. This avoids the requirement of installing OCI CLI on the database node.

Property	Description
Syntax	`DUMPTRANSFERDETAILS_SOURCE_TRANSFERNODE_SUDOPATH = path`
Default value	No default value
Range of values	A string value is expected

DUMPTRANSFERDETAILS_SOURCE_TRANSFERNODE_USER

`DUMPTRANSFERDETAILS_SOURCE_TRANSFERNODE_USER` specifies the user allowed to execute OCI CLI in the dump transfer node.

The `DUMPTRANSFERDETAILS` parameters specify details to transfer dumps from the source node to the target node. Supported modes of transfer are OSS and secure copy. Upload of dumps to OSS can be either through OCI CLI or Curl.

The `DUMPTRANSFERDETAILS_SOURCE` parameters specify details to upload the Dump files from source node Data Pump dump location to Oracle Cloud Object Storage.

The `DUMPTRANSFERDETAILS_SOURCE_TRANSFERNODE` parameters are configured when you use a standalone server as transfer server. Transfer dump files using Oracle Cloud OCI-CLI or curl, or copy to or from a node other than the database server. This option can be leveraged for cases where OCI CLI is installed and configured in a specific node per data center and the node havihas the Data Pump export or import directory path shared with it. This avoids the requirement of installing OCI CLI on the database node.

Property	Description
Syntax	`DUMPTRANSFERDETAILS_SOURCE_TRANSFERNODE_USER = user`
Default value	No default value
Range of values	A string value is expected

DUMPTRANSFERDETAILS_SOURCE_TRANSFERNODE_USERKEY

`DUMPTRANSFERDETAILS_SOURCE_TRANSFERNODE_USER` specifies the user's authentication key.

The `DUMPTRANSFERDETAILS` parameters specify details to transfer dumps from the source node to the target node. Supported modes of transfer are OSS and secure copy. Upload of dumps to OSS can be either through OCI CLI or Curl.

The `DUMPTRANSFERDETAILS_SOURCE` parameters specify details to upload the Dump files from source node Data Pump dump location to Oracle Cloud Object Storage.

The `DUMPTRANSFERDETAILS_SOURCE_TRANSFERNODE` parameters are configured when you use a standalone server as transfer server. Transfer dump files using Oracle Cloud OCI-CLI or curl, or copy to or from a node other than the database server. This option can be leveraged for cases where OCI CLI is installed and configured in a specific node per data center and the node havihas the Data Pump export or import directory path shared with it. This avoids the requirement of installing OCI CLI on the database node.

Property	Description
Syntax	`DUMPTRANSFERDETAILS_SOURCE_TRANSFERNODE_USERKEY = `*`user_auth_key`*
Default value	No default value
Range of values	A string value is expected

DUMPTRANSFERDETAILS_SOURCE_USEOCICLI

`DUMPTRANSFERDETAILS_SOURCE_USEOCICLI` indicates that transfer dump files use Oracle Cloud Infrastructure command line interface (CLI).

The `DUMPTRANSFERDETAILS` parameters specify details to transfer dumps from the source node to the target node. Supported modes of transfer are OSS and secure copy. Upload of dumps to OSS can be either through OCI CLI or Curl.

The `DUMPTRANSFERDETAILS_SOURCE` parameters specify details to upload the Dump files from source node Data Pump dump location to Oracle Cloud Object Storage.

Property	Description
Syntax	`DUMPTRANSFERDETAILS_SOURCE_USEOCICLI = {TRUE	FALSE}`
Default value	`FALSE`
Range of values	`TRUE` or `FALSE`

DUMPTRANSFERDETAILS_TARGET_OCIHOME

`DUMPTRANSFERDETAILS_TARGET_OCIHOME` specifies the Oracle Cloud Infrastructure command line interface (CLI) Binary path.

The `DUMPTRANSFERDETAILS` parameters specify details to transfer dumps from the source node to the target node. Supported modes of transfer are OSS and secure copy. Upload of dumps to OSS can be either through OCI CLI or Curl.

The `DUMPTRANSFERDETAILS_TARGET` parameters specify details to download the Dump files to the target node Data Pump dump location from Oracle Cloud Object Storage.

Property	Description
Syntax	`DUMPTRANSFERDETAILS_TARGET_OCIHOME = *path*`
Default value	No default value
Range of values	A string value is expected

DUMPTRANSFERDETAILS_TARGET_PARTSIZEMB

`DUMPTRANSFERDETAILS_TARGET_PARTSIZEMB` specifies the part size in MB for chunked transfer.

The `DUMPTRANSFERDETAILS` parameters specify details to transfer dumps from the source node to the target node. Supported modes of transfer are OSS and secure copy. Upload of dumps to OSS can be either through OCI CLI or Curl.

The `DUMPTRANSFERDETAILS_TARGET` parameters specify details to download the Dump files to the target node Data Pump dump location from Oracle Cloud Object Storage.

Property	Description
Syntax	`DUMPTRANSFERDETAILS_TARGET_PARTSIZEMB = *integer*`
Default value	`128`
Range of values	An integer value is expected

DUMPTRANSFERDETAILS_TARGET_TRANSFERNODE_DUMPDIRPATH

`DUMPTRANSFERDETAILS_TARGET_TRANSFERNODE_DUMPDIRPATH` specifies the absolute path of the directory to copy or upload dump files.

The `DUMPTRANSFERDETAILS` parameters specify details to transfer dumps from the source node to the target node. Supported modes of transfer are OSS and secure copy. Upload of dumps to OSS can be either through Oracle Cloud Infrastructure command line interface (CLI) or Curl.

The `DUMPTRANSFERDETAILS_TARGET` parameters specify details to download the Dump files to the target node Data Pump dump location from Oracle Cloud Object Storage.

The `DUMPTRANSFERDETAILS_TARGET_TRANSFERNODE` parameters are configured when you use a standalone server as transfer server. Transfer dump files using the OCI CLI or curl, or copy to or from a node other than the database server. This option can be leveraged for cases where OCI CLI is installed and configured in a specific node per data center and the node has the Data Pump export or import directory path shared with it. This avoids the requirement of installing OCI CLI on the database node.

Property	Description
Syntax	`DUMPTRANSFERDETAILS_TARGET_TRANSFERNODE_DUMPDIRPATH = *path*`
Default value	No default value
Range of values	A string value is expected

DUMPTRANSFERDETAILS_TARGET_TRANSFERNODE_HOST

`DUMPTRANSFERDETAILS_TARGET_TRANSFERNODE_HOST` specifies the dump transfer node host name.

The `DUMPTRANSFERDETAILS` parameters specify details to transfer dumps from the source node to the target node. Supported modes of transfer are OSS and secure copy. Upload of dumps to OSS can be either through Oracle Cloud Infrastructure command line interface (CLI) or Curl.

The `DUMPTRANSFERDETAILS_TARGET` parameters specify details to download the Dump files to the target node Data Pump dump location from Oracle Cloud Object Storage.

The `DUMPTRANSFERDETAILS_TARGET_TRANSFERNODE` parameters are configured when you use a standalone server as transfer server. Transfer dump files using OCI CLI or curl, or copy to or from a node other than the database server. This option can be leveraged for cases where OCI CLI is installed and configured in a specific node per data center and the node has the Data Pump export or import directory path shared with it. This avoids the requirement of installing OCI CLI on the database node.

Property	Description
Syntax	`DUMPTRANSFERDETAILS_TARGET_TRANSFERNODE_HOST = *host_name*`
Default value	No default value
Range of values	A string value is expected

DUMPTRANSFERDETAILS_TARGET_TRANSFERNODE_SUDOPATH

`DUMPTRANSFERDETAILS_TARGET_TRANSFERNODE_SUDOPATH` specifies the Sudo path on the dump transfer node.

The `DUMPTRANSFERDETAILS` parameters specify details to transfer dumps from the source node to the target node. Supported modes of transfer are OSS and secure copy. Upload of dumps to OSS can be either through Oracle Cloud Infrastructure command line interface (CLI) or Curl.

The `DUMPTRANSFERDETAILS_TARGET` parameters specify details to download the Dump files to the target node Data Pump dump location from Oracle Cloud Object Storage.

The `DUMPTRANSFERDETAILS_TARGET_TRANSFERNODE` parameters are configured when you use a standalone server as transfer server. Transfer dump files using OCI CLI or curl, or copy to or from a node other than the database server. This option can be leveraged for cases where OCI CLI is installed and configured in a specific node per data center and the node has the Data Pump export or import directory path shared with it. This avoids the requirement of installing OCI CLI on the database node.

Property	Description
Syntax	`DUMPTRANSFERDETAILS_TARGET_TRANSFERNODE_SUDOPATH = `*`path`*
Default value	No default value
Range of values	A string value is expected

DUMPTRANSFERDETAILS_TARGET_TRANSFERNODE_USER

`DUMPTRANSFERDETAILS_TARGET_TRANSFERNODE_USER` specifies the user allowed to execute OCI CLI in the dump transfer node.

The `DUMPTRANSFERDETAILS` parameters specify details to transfer dumps from the source node to the target node. Supported modes of transfer are OSS and secure copy. Upload of dumps to OSS can be either through Oracle Cloud Infrastructure command line interface (CLI) or Curl.

The `DUMPTRANSFERDETAILS_TARGET` parameters specify details to download the Dump files to the target node Data Pump dump location from Oracle Cloud Object Storage.

The `DUMPTRANSFERDETAILS_TARGET_TRANSFERNODE` parameters are configured when you use a standalone server as transfer server. Transfer dump files using OCI CLI or curl, or copy to or from a node other than the database server. This option can be leveraged for cases where OCI CLI is installed and configured in a specific node per data center and the node has the Data Pump export or import directory path shared with it. This avoids the requirement of installing OCI CLI on the database node.

Property	Description
Syntax	`DUMPTRANSFERDETAILS_TARGET_TRANSFERNODE_USER = `*`user`*
Default value	No default value
Range of values	A string value is expected

DUMPTRANSFERDETAILS_TARGET_TRANSFERNODE_USERKEY

`DUMPTRANSFERDETAILS_TARGET_TRANSFERNODE_USER` specifies the user's authentication key.

The `DUMPTRANSFERDETAILS` parameters specify details to transfer dumps from the source node to the target node. Supported modes of transfer are OSS and secure copy. Upload of dumps to OSS can be either through Oracle Cloud Infrastructure command line interface (CLI) or Curl.

The `DUMPTRANSFERDETAILS_TARGET` parameters specify details to download the Dump files to the target node Data Pump dump location from Oracle Cloud Object Storage.

The `DUMPTRANSFERDETAILS_TARGET_TRANSFERNODE` parameters are configured when you use a standalone server as transfer server. Transfer dump files using OCI CLI or curl, or copy to or from a node other than the database server. This option can be leveraged for cases where OCI CLI is installed and configured in a specific node per data center and the node has the Data Pump export or import directory path shared with it. This avoids the requirement of installing OCI CLI on the database node.

Property	Description
Syntax	`DUMPTRANSFERDETAILS_TARGET_TRANSFERNODE_USERKEY = *user_auth_key*`
Default value	No default value
Range of values	A string value is expected

DUMPTRANSFERDETAILS_TARGET_USEOCICLI

`DUMPTRANSFERDETAILS_TARGET_USEOCICLI` indicates that transfer dump files use Oracle Cloud Infrastructure command line interface (CLI).

The `DUMPTRANSFERDETAILS` parameters specify details to transfer dumps from the source node to the target node. Supported modes of transfer are OSS and secure copy. Upload of dumps to OSS can be either through OCI CLI or Curl.

The `DUMPTRANSFERDETAILS_TARGET` parameters specify details to download the Dump files to the target node Data Pump dump location from Oracle Cloud Object Storage.

Property	Description
Syntax	`DUMPTRANSFERDETAILS_TARGET_USEOCICLI = {TRUE | FALSE}`
Default value	`FALSE`
Range of values	`TRUE` or `FALSE`

DUMPTRANSFERDETAILS_TRANSFERTARGET_DUMPDIRPATH

`DUMPTRANSFERDETAILS_TRANSFERTARGET_DUMPDIRPATH` specifies the absolute path of the directory to copy or upload dump files.

The `DUMPTRANSFERDETAILS` parameters specify details to transfer dumps from the source node to the target node. Supported modes of transfer are OSS and secure copy. Upload of dumps to OSS can be either through OCI CLI or Curl.

The `DUMPTRANSFERDETAILS_TRANSFERTARGET` parameters specify details for the target node to copy the Dump files from source node Data Pump dump location or transfer node directory path specified. Default target database host.

Property	Description
Syntax	`DUMPTRANSFERDETAILS_TRANSFERTARGET_DUMPDIRPATH = *path*`
Default value	No default value
Range of values	A string value is expected

DUMPTRANSFERDETAILS_TRANSFERTARGET_HOST

`DUMPTRANSFERDETAILS_TRANSFERTARGET_HOST` specifies the dump transfer node host name.

The `DUMPTRANSFERDETAILS` parameters specify details to transfer dumps from the source node to the target node. Supported modes of transfer are OSS and secure copy. Upload of dumps to OSS can be either through OCI CLI or Curl.

The `DUMPTRANSFERDETAILS_TRANSFERTARGET` parameters specify details for the target node to copy the Dump files from source node Data Pump dump location or transfer node directory path specified. Default target database host.

Property	Description
Syntax	`DUMPTRANSFERDETAILS_TRANSFERTARGET_HOST = *host_name*`
Default value	No default value
Range of values	A string value is expected

DUMPTRANSFERDETAILS_TRANSFERTARGET_SUDOPATH

`DUMPTRANSFERDETAILS_TRANSFERTARGET_SUDOPATH` specifies the Sudo path on the dump transfer node.

The `DUMPTRANSFERDETAILS` parameters specify details to transfer dumps from the source node to the target node. Supported modes of transfer are OSS and secure copy. Upload of dumps to OSS can be either through OCI CLI or Curl.

The `DUMPTRANSFERDETAILS_TRANSFERTARGET` parameters specify details for the target node to copy the Dump files from source node Data Pump dump location or transfer node directory path specified. Default target database host.

Property	Description
Syntax	`DUMPTRANSFERDETAILS_TRANSFERTARGET_SUDOPATH = *path*`
Default value	No default value
Range of values	A string value is expected

DUMPTRANSFERDETAILS_TRANSFERTARGET_USER

`DUMPTRANSFERDETAILS_TRANSFERTARGET_USER` specifies the user allowed to execute OCI CLI in specified node.

The `DUMPTRANSFERDETAILS` parameters specify details to transfer dumps from the source node to the target node. Supported modes of transfer are OSS and secure copy. Upload of dumps to OSS can be either through OCI CLI or Curl.

The `DUMPTRANSFERDETAILS_TRANSFERTARGET` parameters specify details for the target node to copy the Dump files from source node Data Pump dump location or transfer node directory path specified. Default target database host.

Property	Description
Syntax	`DUMPTRANSFERDETAILS_TRANSFERTARGET_USER = *user*`
Default value	`There is no default value`
Range of values	An integer value is expected

DUMPTRANSFERDETAILS_TRANSFERTARGET_USERKEY

`DUMPTRANSFERDETAILS_TRANSFERTARGET_USERKEY` specifies the user's authentication key.

The `DUMPTRANSFERDETAILS` parameters specify details to transfer dumps from the source node to the target node. Supported modes of transfer are OSS and secure copy. Upload of dumps to OSS can be either through OCI CLI or Curl.

The `DUMPTRANSFERDETAILS_TRANSFERTARGET` parameters specify details for the target node to copy the Dump files from source node Data Pump dump location or transfer node directory path specified. Default target database host.

Property	Description
Syntax	`DUMPTRANSFERDETAILS_TRANSFERTARGET_USERKEY = *user_auth_key*`
Default value	No default value
Range of values	A string value is expected

EXCLUDEOBJECTS-*LIST_ELEMENT_NUMBER*

`EXCLUDEOBJECTS-LIST_ELEMENT_NUMBER` specifies database objects to exclude from migration.

To exclude multiple objects, increment the integer appended to the parameter name, as shown in the examples below.

```
EXCLUDEOBJECTS-1=owner:ownerValue1st, objectName:objectNameValue1st
EXCLUDEOBJECTS-1=owner:ownerValue2nd, objectName:objectNameValue2nd
```

Property	Description
Syntax	`EXCLUDEOBJECTS-LIST_ELEMENT_NUMBER = owner:ownerValue, objectName:objectNameValue,`
Default value	There is no default value.
Range of values	`owner` can be the `OWNER` value of a database object `objectName` can be the `OBJECT_NAME` value of a database object. Only `TABLE` object types are supported for exlusion by Database Migration. You can filter owner and object values by any valid pattern in Java class Pattern. For example, you can enter `.*` in the `objectName` argument to exclude all tables (incluing views) owned by `owner`.

GOLDENGATEHUB_ADMINUSERNAME

For online logical migration, you must set the `GOLDENGATEHUB` parameters to provide Zero Downtime Migration with details about Oracle GoldenGate Microservices configuration. `GOLDENGATEHUB_ADMINUSERNAME` specifies the GoldenGate hub administrator user name.

Property	Description
Syntax	`GOLDENGATEHUB_ADMINUSERNAME = user_name`
Default value	There is no default value
Range of values	Oracle GoldenGate hub's administrator user name

GOLDENGATEHUB_COMPUTEID

For online logical migration, you must set the `GOLDENGATEHUB` parameters to provide ZDM with details about Oracle GoldenGate Microservices configuration. `GOLDENGATEHUB_COMPUTEID` specifies the Oracle Cloud identifier of the VM.

Property	Description
Syntax	`GOLDENGATEHUB_COMPUTEID = vm_identifier`
Default value	There is no default value
Range of values	Oracle Cloud identifier of the VM

GOLDENGATEHUB_SOURCEDEPLOYMENTNAME

For online logical migration, you must set the `GOLDENGATEHUB` parameters to provide ZDM with details about Oracle GoldenGate Microservices configuration. `GOLDENGATEHUB_SOURCEDEPLOYMENTNAME` specifies the name of the GoldenGate Microservices deployment to operate on the source database.

Property	Description
Syntax	`GOLDENGATEHUB_SOURCEDEPLOYMENTNAME = GG_microservices_deployment_name`
Default value	There is no default value
Range of values	Name of the GoldenGate Microservices deployment to operate on the source database

GOLDENGATEHUB_TARGETDEPLOYMENTNAME

For online logical migration, you must set the `GOLDENGATEHUB` parameters to provide ZDM with details about Oracle GoldenGate Microservices configuration. `GOLDENGATEHUB_TARGETDEPLOYMENTNAME` specifies the name of the GoldenGate Microservices deployment to operate on the target database.

Property	Description
Syntax	`GOLDENGATEHUB_TARGETDEPLOYMENTNAME = GG_microservices_deployment_name`
Default value	There is no default value
Range of values	Name of the GoldenGate Microservices deployment to operate on the target database

GOLDENGATEHUB_URL

For online logical migration, you must set the `GOLDENGATEHUB` parameters to provide ZDM with details about Oracle GoldenGate Microservices configuration. `GOLDENGATEHUB_URL` specifies the Oracle GoldenGate hub's REST endpoint.

Property	Description
Syntax	`GOLDENGATEHUB_URL = GG_hub_rest_endpoint`

Property	Description
Default value	There is no default value
Range of values	Oracle GoldenGate hub's REST endpoint

GOLDENGATESETTINGS_ACCEPTABLELAG

For online logical migration, you can set the optional `GOLDENGATESETTINGS` parameters to provide ZDM with details about Oracle GoldenGate Microservices configuration. ZDM will monitor GoldenGate end-to-end latency until the lag time is lower than the specified value, in seconds, in `GOLDENGATESETTINGS_ACCEPTABLELAG`.

Property	Description
Syntax	`GOLDENGATESETTINGS_ACCEPTABLELAG = seconds`
Default value	30 seconds

GOLDENGATESETTINGS_EXTRACT_NAME

`GOLDENGATESETTINGS_EXTRACT_NAME` specifies a name for the Extract processes.

For online logical migration, you can set the optional `GOLDENGATESETTINGS` parameters to provide ZDM with details about Oracle GoldenGate Microservices configuration.

The `GOLDENGATESETTINGS_EXTRACT` parameters define settings for the Integrated Extract process. Only one Extract can be configured.

Property	Description
Syntax	`GOLDENGATESETTINGS_EXTRACT_NAME = nameValue`
Default value	There is no default value
Valid values	A string value is expected

GOLDENGATESETTINGS_EXTRACT_PERFORMANCEPROFILE

`GOLDENGATESETTINGS_EXTRACT_PERFORMANCEPROFILE` tunes Integrated Capture. Use this setting to automatically configure relevant parameters to achieve the desired throughput and latency. This parameter is optional.

For online logical migration, you can set the optional `GOLDENGATESETTINGS` parameters to provide ZDM with details about Oracle GoldenGate Microservices configuration.

The `GOLDENGATESETTINGS_EXTRACT` parameters define settings for the Integrated Extract process. Only one Extract can be configured.

Property	Description
Syntax	`GOLDENGATESETTINGS_EXTRACT_PERFORMANCEPROFILE = [HIGH | MEDIUM | LOW-RES]`
Default value	`MEDIUM`
Valid values	`HIGH`, `MEDIUM`, `LOW-RES`

GOLDENGATESETTINGS_EXTRACT_WARNLONGTRANS_CHECKINTERVAL

`GOLDENGATESETTINGS_EXTRACT_WARNLONGTRANS_CHECKINTERVAL` specifies the frequency in seconds with which Oracle GoldenGate checks for long-running transactions. This parameter is optional.

For online logical migration, you can set the optional `GOLDENGATESETTINGS` parameters to provide ZDM with details about Oracle GoldenGate Microservices configuration.

The `GOLDENGATESETTINGS_EXTRACT` parameters define settings for the Integrated Extract process. Only one Extract can be configured.

`GOLDENGATESETTINGS_EXTRACT_WARNLONGTRANS` parameters specify a length of time in seconds that a transaction can be open before Extract generates a warning message that the transaction is long-running.

Property	Description
Syntax	`GOLDENGATESETTINGS_EXTRACT_WARNLONGTRANS_CHECKINTERVAL =` *`integer`*
Default value	There is no default value
Valid values	An integer value is expected

GOLDENGATESETTINGS_EXTRACT_WARNLONGTRANS_DURATION

`GOLDENGATESETTINGS_EXTRACT_WARNLONGTRANS_DURATION` specifies a length of time in seconds that a transaction can be open before Extract generates a warning message that the transaction is long-running. This parameter is optional.

For online logical migration, you can set the optional `GOLDENGATESETTINGS` parameters to provide ZDM with details about Oracle GoldenGate Microservices configuration.

The `GOLDENGATESETTINGS_EXTRACT` parameters define settings for the Integrated Extract process. Only one Extract can be configured.

`GOLDENGATESETTINGS_EXTRACT_WARNLONGTRANS` parameters specify a length of time in seconds that a transaction can be open before Extract generates a warning message that the transaction is long-running.

Property	Description
Syntax	`GOLDENGATESETTINGS_EXTRACT_WARNLONGTRANS_DURATION = *integer*`
Default value	There is no default value
Valid values	An integer value is expected

GOLDENGATESETTINGS_REPLICAT_MAPPARALLELISM

`GOLDENGATESETTINGS_REPLICAT_MAPPARALLELISM` specifies the number of threads used to read trail files (valid for Parallel Replicat). This parameter is optional.

For online logical migration, you can set the optional `GOLDENGATESETTINGS` parameters to provide ZDM with details about Oracle GoldenGate Microservices configuration.

The `GOLDENGATESETTINGS_REPLICAT` parameters define settings for the Parallel Replicat process.

Property	Description
Syntax	`GOLDENGATESETTINGS_REPLICAT_MAPPARALLELISM = *integer*`
Default value	4
Valid values	An integer value is expected

GOLDENGATESETTINGS_REPLICAT_MAXAPPLYPARALLELISM

`GOLDENGATESETTINGS_REPLICAT_MAXAPPLYPARALLELISM` defines the range in which the Replicat automatically adjusts its apply parallelism (valid for Parallel Replicat). This parameter is optional.

For online logical migration, you can set the optional `GOLDENGATESETTINGS` parameters to provide ZDM with details about Oracle GoldenGate Microservices configuration.

The `GOLDENGATESETTINGS_REPLICAT` parameters define settings for the Parallel Replicat process.

Property	Description
Syntax	`GOLDENGATESETTINGS_REPLICAT_MAXAPPLYPARALLELISM = *integer*`
Default value	50
Valid values	An integer value is expected

GOLDENGATESETTINGS_REPLICAT_MINAPPLYPARALLELISM

`GOLDENGATESETTINGS_REPLICAT_MINAPPLYPARALLELISM` defines the range in which the Replicat automatically adjusts its apply parallelism (valid for Parallel Replicat). This parameter is optional.

For online logical migration, you can set the optional `GOLDENGATESETTINGS` parameters to provide ZDM with details about Oracle GoldenGate Microservices configuration.

The `GOLDENGATESETTINGS_REPLICAT` parameters define settings for the Parallel Replicat process.

Property	Description
Syntax	`GOLDENGATESETTINGS_REPLICAT_MINAPPLYPARALLELISM = *integer*`
Default value	4
Valid values	An integer value is expected

GOLDENGATESETTINGS_REPLICAT_NAME

`GOLDENGATESETTINGS_REPLICAT_NAME` specifies a name for the Replicat process. This parameter is optional.

For online logical migration, you can set the optional `GOLDENGATESETTINGS` parameters to provide ZDM with details about Oracle GoldenGate Microservices configuration.

The `GOLDENGATESETTINGS_REPLICAT` parameters define settings for the Parallel Replicat process.

Property	Description
Syntax	`GOLDENGATESETTINGS_REPLICAT_NAME = *nameValue*`
Default value	There is no default value
Valid values	A string value is expected

INCLUDEOBJECTS-*LIST_ELEMENT_NUMBER*

`INCLUDEOBJECTS-*LIST_ELEMENT_NUMBER*` specifies database objects to include for migration.

To include multiple objects, increment the integer appended to the parameter name, as shown in the examples below.

```
INCLUDEOBJECTS-1=owner:ownerValue1st, objectName:objectNameValue1st
INCLUDEOBJECTS-1=owner:ownerValue2nd, objectName:objectNameValue2nd
```

Property	Description
Syntax	`INCLUDEOBJECTS-`*`LIST_ELEMENT_NUMBER`*` = owner:`*`ownerValue`*`, objectName:`*`objectNameValue`*
Default value	There is no default value
Range of values	An entry specifying the owner and object name is expected, as shown in the examples above.

MIGRATION_TYPE

In a logical migration, the required `MIGRATION_TYPE` parameter specifies whether the migration is online or offline.

Property	Description
Syntax	`MIGRATION_TYPE = {ONLINE_LOGICAL | OFFLINE_LOGICAL}`
Default value	`OFFLINE_LOGICAL`
Range of values	`ONLINE_LOGICAL` `OFFLINE_LOGICAL`

OCIAUTHENTICATIONDETAILS_REGIONID

To call REST APIs, you must configure the `OCIAUTHENTICATIONDETAILS` parameters. `OCIAUTHENTICATIONDETAILS_REGIONID` specifies the OCI region identifier.

See the Region Identifier column in the table at Regions and Availability Domains.

Property	Description
Syntax	`OCIAUTHENTICATIONDETAILS_REGIONID = `*`region_id`*
Default value	There is no default value

OCIAUTHENTICATIONDETAILS_USERPRINCIPAL_FINGERPRINT

To call REST APIs, you must configure the `OCIAUTHENTICATIONDETAILS` parameters. `OCIAUTHENTICATIONDETAILS_USERPRINCIPAL_FINGERPRINT` specifies the fingerprint of the public API key.

See Required Keys and OCIDs for more information.

Property	Description
Syntax	`OCIAUTHENTICATIONDETAILS_USERPRINCIPAL_FINGERPRINT = `*`fingerprint`*

Property	Description
Default value	There is no default value

OCIAUTHENTICATIONDETAILS_USERPRINCIPAL_PRIVATEKEYFILE

To call REST APIs, you must configure the `OCIAUTHENTICATIONDETAILS` parameters. `OCIAUTHENTICATIONDETAILS_USERPRINCIPAL_PRIVATEKEYFILE` specifies the absolute path of API private key file.

See Required Keys and OCIDs for more information.

Property	Description
Syntax	`OCIAUTHENTICATIONDETAILS_USERPRINCIPAL_PRIVATEKEYFILE = `*`path`*
Default value	There is no default value

OCIAUTHENTICATIONDETAILS_USERPRINCIPAL_TENANTID

To call OCI REST APIs, you must configure the `OCIAUTHENTICATIONDETAILS` parameters. `OCIAUTHENTICATIONDETAILS_USERPRINCIPAL_TENANTID` specifies the OCID of the OCI tenancy.

You can find the tenant OCID on OCI at Governance and Administration, Administration, Tenancy Details. The tenancy OCID is shown under Tenancy Information.

Example:
ocid1.tenancy.oc1..aaaaaaaaba3pv6wkcr4jqae5f44n2b2m2yt2j6rx32uzr4h25vqstifsfds
q

See Managing the Tenancy and Required Keys and OCIDs for more information.

Property	Description
Syntax	`OCIAUTHENTICATIONDETAILS_USERPRINCIPAL_TENANTID = `*`ocid_string`*
Default value	There is no default value
Valid values	A string value is expected

OCIAUTHENTICATIONDETAILS_USERPRINCIPAL_USERID

To call OCI REST APIs, you must configure the `OCIAUTHENTICATIONDETAILS` parameters. `OCIAUTHENTICATIONDETAILS_USERPRINCIPAL_USERID` specifies the OCID of the IAM user.

You can find the IAM OCID on OCI at Console, Profile, User Settings.

See Managing Users and Required Keys and OCIDs for more information.

Property	Description
Syntax	`OCIAUTHENTICATIONDETAILS_USERPRINCIPAL_USERID = ` *`userid`*
Default value	There is no default value

OCIPROXY_HOSTNAME

The `OCIPROXY` parameters specify details about the proxy for connecting to OCI REST endpoints. `OCIPROXY_HOSTNAME` specifies the HTTP proxy host name.

Property	Description
Syntax	`OCIPROXY_HOSTNAME = ` *`hostname`*
Default value	There is no default value

OCIPROXY_PORT

The `OCIPROXY` parameters specify details about the proxy for connecting to OCI REST endpoints. `OCIPROXY_PORT` specifies the HTTP proxy port number.

Property	Description
Syntax	`OCIPROXY_PORT = ` *`port number`*
Default value	There is no default value

SOURCECONTAINERDATABASE_ADMINUSERNAME

For online logical migrations, the `SOURCECONTAINERDATABASE` parameters specify connection details for the source container root. `SOURCECONTAINERDATABASE_ADMINUSERNAME` specifies the database administrator user name.

Property	Description
Syntax	`SOURCECONTAINERDATABASE_ADMINUSERNAME = ` *`username`*
Default value	There is no default value

SOURCECONTAINERDATABASE_CONNECTIONDETAILS_BASTIONDETAILS_IDENTITYFILE

For online logical migrations, the `SOURCECONTAINERDATABASE` parameters specify connection details for the source container root. `SOURCECONTAINERDATABASE_CONNECTIONDETAILS_BASTIONDETAILS_IDENTITYFILE` specifies the identity file to access bastion, as part of the database connection details for bastion-based access to database.

Property	Description
Syntax	`SOURCECONTAINERDATABASE_CONNECTIONDETAILS_BASTIONDETAILS_IDENTITYFILE = *bastion_id_file*`
Default value	There is no default value

SOURCECONTAINERDATABASE_CONNECTIONDETAILS_BASTIONDETAILS_IP

For online logical migrations, the `SOURCECONTAINERDATABASE` parameters specify connection details for the source container root. `SOURCECONTAINERDATABASE_CONNECTIONDETAILS_BASTIONDETAILS_IP` specifies the IP addressof the bastion host, as part of the database connection details for bastion-based access to database.

Property	Description
Syntax	`SOURCECONTAINERDATABASE_CONNECTIONDETAILS_BASTIONDETAILS_IP = *bastion_ip_address*`
Default value	There is no default value

SOURCECONTAINERDATABASE_CONNECTIONDETAILS_BASTIONDETAILS_PORT

For online logical migrations, the `SOURCECONTAINERDATABASE` parameters specify connection details for the source container root. `SOURCECONTAINERDATABASE_CONNECTIONDETAILS_BASTIONDETAILS_PORT` specifies the bastion host port detail, as part of the database connection details for bastion-based access to database.

Property	Description
Syntax	`SOURCECONTAINERDATABASE_CONNECTIONDETAILS_BASTIONDETAILS_PORT =` *`bastion_port_number`*
Default value	There is no default value

SOURCECONTAINERDATABASE_CONNECTIONDETAILS_BASTIONDETAILS_REMOTEHOSTIP

For online logical migrations, the `SOURCECONTAINERDATABASE` parameters specify connection details for the source container root. `SOURCECONTAINERDATABASE_CONNECTIONDETAILS_BASTIONDETAILS_REMOTEHOSTIP` specifies the remote host IP address to access from the bastion, as part of the database connection details for bastion-based access to database.

Property	Description
Syntax	`SOURCECONTAINERDATABASE_CONNECTIONDETAILS_BASTIONDETAILS_REMOTEHOSTIP =` *`ip_address`*
Default value	There is no default value

SOURCECONTAINERDATABASE_CONNECTIONDETAILS_BASTIONDETAILS_USERNAME

For online logical migrations, the `SOURCECONTAINERDATABASE` parameters specify connection details for the source container root. `SOURCECONTAINERDATABASE_CONNECTIONDETAILS_BASTIONDETAILS_USERNAME` specifies the user name to access the bastion, as part of the database connection details for bastion-based access to database.

Property	Description
Syntax	`SOURCECONTAINERDATABASE_CONNECTIONDETAILS_BASTIONDETAILS_USERNAME =` *`username`*
Default value	There is no default value

SOURCECONTAINERDATABASE_CONNECTIONDETAILS_HOST

For online logical migrations, the `SOURCECONTAINERDATABASE` parameters specify connection details for the source container root. `SOURCECONTAINERDATABASE_CONNECTIONDETAILS_HOST` specifies the listener host name or IP address. Not required for migrations to Autonomous Database.

Property	Description
Syntax	`SOURCECONTAINERDATABASE_CONNECTIONDETAILS_HOST = listener_host`
Default value	There is no default value

SOURCECONTAINERDATABASE_CONNECTIONDETAILS_PORT

For online logical migrations, the `SOURCECONTAINERDATABASE` parameters specify connection details for the source container root. `SOURCECONTAINERDATABASE_CONNECTIONDETAILS_PORT` specifies the listener port number. Not required for migrations to Autonomous Database.

Property	Description
Syntax	`SOURCECONTAINERDATABASE_CONNECTIONDETAILS_PORT = listener_port`
Default value	There is no default value

SOURCECONTAINERDATABASE_CONNECTIONDETAILS_PROXYDETAILS_HOSTNAME

For online logical migrations, the `SOURCECONTAINERDATABASE_CONNECTIONDETAILS_PROXYDETAILS` parameters specify connection details for the source container root through an HTTP proxy. `SOURCECONTAINERDATABASE_CONNECTIONDETAILS_PROXYDETAILS_HOSTNAME` specifies the HTTPS proxy host name.

Property	Description
Syntax	`SOURCECONTAINERDATABASE_CONNECTIONDETAILS_PROXYDETAILS_HOSTNAME = proxy_hostname`
Default value	There is no default value

SOURCECONTAINERDATABASE_CONNECTIONDETAILS_PROXYDETAILS_PORT

For online logical migrations, the `SOURCECONTAINERDATABASE_CONNECTIONDETAILS_PROXYDETAILS` parameters specify connection details for the source container root through an HTTP proxy.

`SOURCECONTAINERDATABASE_CONNECTIONDETAILS_PROXYDETAILS_PORT` specifies the HTTPS proxy host port number.

Property	Description
Syntax	`SOURCECONTAINERDATABASE_CONNECTIONDETAILS_PROXYDETAILS_PORT =` *`proxy_port`*
Default value	There is no default value

SOURCECONTAINERDATABASE_CONNECTIONDETAILS_SERVICENAME

For online logical migrations, the `SOURCECONTAINERDATABASE` parameters specify connection details for the source container root.
`SOURCECONTAINERDATABASE_CONNECTIONDETAILS_SERVICENAME` specifies the fully qualified service name. Not required for Autonomous Database.

Property	Description
Syntax	`SOURCECONTAINERDATABASE_CONNECTIONDETAILS_SERVICENAME =` *`service_name`*
Default value	There is no default value

SOURCECONTAINERDATABASE_CONNECTIONDETAILS_TLSDETAILS_CREDENTIALSLOCATION

For online logical migrations, the `SOURCECONTAINERDATABASE_CONNECTIONDETAILS_TLSDETAILS` parameters specify details for TLS connection to the database. These settings are not required if you are using TCP.
`SOURCECONTAINERDATABASE_CONNECTIONDETAILS_TLSDETAILS_CREDENTIALSLOCATION` specifies the directory containing client credentials (wallet, keystore, trustfile, etc.). Not required for Autonomous Database.

Property	Description
Syntax	`SOURCECONTAINERDATABASE_CONNECTIONDETAILS_TLSDETAILS_CREDENTIALSLOCATION =` *`directory`*
Default value	There is no default value

SOURCECONTAINERDATABASE_CONNECTIONDETAILS_TLSDETAILS_DISTINGUISHEDNAME

For online logical migrations, the `SOURCECONTAINERDATABASE_CONNECTIONDETAILS_TLSDETAILS` parameters specify details for

TLS connection to the database. These settings are not required if you are using TCP. `SOURCECONTAINERDATABASE_CONNECTIONDETAILS_TLSDETAILS_DISTINGUISHEDNAME` specifies the distinguished name (DN) of the database server (`SSL_SERVER_CERT_DN`). Not required for Autonomous Database.

Property	Description
Syntax	`SOURCECONTAINERDATABASE_CONNECTIONDETAILS_TLSDETAILS_DISTINGUISHEDNAME =` *`distinguished_name`*
Default value	There is no default value

SOURCECONTAINERDATABASE_GGADMINUSERNAME

For online logical migrations, the `SOURCECONTAINERDATABASE` parameters specify connection details for the source container root. `SOURCECONTAINERDATABASE_GGADMINUSERNAME` specifies the GoldenGate administrator username.

Property	Description
Syntax	`SOURCECONTAINERDATABASE_GGADMINUSERNAME =` *`GoldenGate administrator username`*
Default value	There is no default value

SOURCEDATABASE_ADMINUSERNAME

The `SOURCEDATABASE` parameters specify connection details for the source database. `SOURCEDATABASE_ADMINUSERNAME` specifies the source database administrator user name.

Property	Description
Syntax	`SOURCEDATABASE_ADMINUSERNAME =` *`source administrator username`*
Default value	There is no default value

SOURCEDATABASE_CONNECTIONDETAILS_BASTIONDETAILS_IDENTITYFILE

The `SOURCEDATABASE` parameters specify connection details for the source database. For bastion-based access to the database, `SOURCEDATABASE_CONNECTIONDETAILS_BASTIONDETAILS_IDENTITYFILE` specifies the identity file to access the bastion.

Property	Description
Syntax	`SOURCEDATABASE_CONNECTIONDETAILS_BASTIONDETAILS_IDENTITYFILE = `*`identity file`*
Default value	There is no default value

SOURCEDATABASE_CONNECTIONDETAILS_BASTIONDETAILS_IP

The `SOURCEDATABASE` parameters specify connection details for the source database. For bastion-based access to the database, `SOURCEDATABASE_CONNECTIONDETAILS_BASTIONDETAILS_IP` specifies the IP address of the bastion host.

Property	Description
Syntax	`SOURCEDATABASE_CONNECTIONDETAILS_BASTIONDETAILS_IP = `*`ip address`*
Default value	There is no default value

SOURCEDATABASE_CONNECTIONDETAILS_BASTIONDETAILS_PORT

The `SOURCEDATABASE` parameters specify connection details for the source database. For bastion-based access to the database, `SOURCEDATABASE_CONNECTIONDETAILS_BASTIONDETAILS_PORT` specifies the port number of the bastion host.

Property	Description
Syntax	`SOURCEDATABASE_CONNECTIONDETAILS_BASTIONDETAILS_PORT = `*`port_number`*
Default value	There is no default value

SOURCEDATABASE_CONNECTIONDETAILS_BASTIONDETAILS_REMOTEHOSTIP

The `SOURCEDATABASE` parameters specify connection details for the source database. For bastion-based access to the database, `SOURCEDATABASE_CONNECTIONDETAILS_BASTIONDETAILS_REMOTEHOSTIP` specifies the remote host IP address to access from the bastion.

Property	Description
Syntax	`SOURCEDATABASE_CONNECTIONDETAILS_BASTIONDETAILS_REMOTEHOSTIP = ip address`
Default value	There is no default value

SOURCEDATABASE_CONNECTIONDETAILS_BASTIONDETAILS_USERNAME

The `SOURCEDATABASE` parameters specify connection details for the source database. For bastion-based access to the database, `SOURCEDATABASE_CONNECTIONDETAILS_BASTIONDETAILS_USERNAME` specifies the user name to access the bastion.

Property	Description
Syntax	`SOURCEDATABASE_CONNECTIONDETAILS_BASTIONDETAILS_USERNAME = username`
Default value	There is no default value

SOURCEDATABASE_CONNECTIONDETAILS_HOST

The `SOURCEDATABASE` parameters specify connection details for the source database. `SOURCEDATABASE_CONNECTIONDETAILS_HOST` specifies the listener host name or IP address. Not required for Autonomous Database.

Property	Description
Syntax	`SOURCEDATABASE_CONNECTIONDETAILS_HOST = hostname_or_ip`
Default value	There is no default value

SOURCEDATABASE_CONNECTIONDETAILS_PORT

The `SOURCEDATABASE` parameters specify connection details for the source database. `SOURCEDATABASE_CONNECTIONDETAILS_PORT` specifies the listener port number. Not required for Autonomous Database.

Property	Description
Syntax	`SOURCEDATABASE_CONNECTIONDETAILS_PORT = listener port number`
Default value	There is no default value

SOURCEDATABASE_CONNECTIONDETAILS_PROXYDETAILS_HOSTNAME

The `SOURCEDATABASE` parameters specify connection details for the source database. For connecting to the source database through an HTTPS proxy, `SOURCEDATABASE_CONNECTIONDETAILS_PROXYDETAILS_HOSTNAME` specifies the proxy host name.

Property	Description
Syntax	`SOURCEDATABASE_CONNECTIONDETAILS_PROXYDETAILS_HOSTNAME = `*`proxy host name`*
Default value	There is no default value

SOURCEDATABASE_CONNECTIONDETAILS_PROXYDETAILS_PORT

The `SOURCEDATABASE` parameters specify connection details for the source database. For connecting to the source database through an HTTPS proxy, `SOURCEDATABASE_CONNECTIONDETAILS_PROXYDETAILS_PORT` specifies the HTTP proxy port number.

Property	Description
Syntax	`SOURCEDATABASE_CONNECTIONDETAILS_PROXYDETAILS_PORT = `*`proxy port number`*
Default value	There is no default value

SOURCEDATABASE_CONNECTIONDETAILS_SERVICENAME

The `SOURCEDATABASE` parameters specify connection details for the source database. `SOURCEDATABASE_CONNECTIONDETAILS_SERVICENAME` specifies the fully qualified service name. Not required for Autonomous Database.

Property	Description
Syntax	`SOURCEDATABASE_CONNECTIONDETAILS_SERVICENAME = `*`service name`*
Default value	There is no default value

SOURCEDATABASE_CONNECTIONDETAILS_TLSDETAILS_CREDENTIALSLOCATION

The `SOURCEDATABASE` parameters specify connection details for the source database. For a TLS connection to the database, set `SOURCEDATABASE_CONNECTIONDETAILS_TLSDETAILS_CREDENTIALSLOCATION`, which specifies the directory containing client credentials (wallet, keystore, trustfile, etc.) Not required if using TCP. Not required for Autonomous Database.

Property	Description
Syntax	`SOURCEDATABASE_CONNECTIONDETAILS_TLSDETAILS_CREDENTIALSLOCATION =` *`directory`*
Default value	There is no default value

SOURCEDATABASE_CONNECTIONDETAILS_TLSDETAILS_DISTINGUISHEDNAME

The `SOURCEDATABASE` parameters specify connection details for the source database. For a TLS connection to the database, set `SOURCEDATABASE_CONNECTIONDETAILS_TLSDETAILS_DISTINGUISHEDNAME`, which specifies the distinguished name (DN) of the database server (`SSL_SERVER_CERT_DN`). Not required if using TCP. Not required for Autonomous Database.

Property	Description
Syntax	`SOURCEDATABASE_CONNECTIONDETAILS_TLSDETAILS_DISTINGUISHEDNAME =` *`distinguished_name`*
Default value	There is no default value

SOURCEDATABASE_GGADMINUSERNAME

The `SOURCEDATABASE` parameters specify connection details for the source database. For online logical migrations, `SOURCEDATABASE_GGADMINUSERNAME` specifies the GoldenGate administrator user name.

Property	Description
Syntax	`SOURCEDATABASE_GGADMINUSERNAME =` *`gg_admin_username`*
Default value	There is no default value

TARGETDATABASE_ADMINUSERNAME

The `TARGETDATABASE` parameters specify connection details for the target OCI database. `TARGETDATABASE_ADMINUSERNAME` specifies the database administrator user name.

Property	Description
Syntax	`TARGETDATABASE_ADMINUSERNAME = `*`username`*
Default value	There is no default value

TARGETDATABASE_CONNECTIONDETAILS_BASTIONDETAILS_IDENTITYFILE

The `TARGETDATABASE_CONNECTIONDETAILS` parameters specify connection details for the target OCI database. These are optional for Autonomous Database; however if an HTTP proxy is required to connect, specify them. For bastion-based access to database, `TARGETDATABASE_CONNECTIONDETAILS_BASTIONDETAILS_IDENTITYFILE` specifies the identity file to access the bastion.

Property	Description
Syntax	`TARGETDATABASE_CONNECTIONDETAILS_BASTIONDETAILS_IDENTITYFILE = `*`bastion_id_file`*
Default value	There is no default value

TARGETDATABASE_CONNECTIONDETAILS_BASTIONDETAILS_IP

The `TARGETDATABASE_CONNECTIONDETAILS` parameters specify connection details for the target OCI database. These are optional for Autonomous Database; however if an HTTP proxy is required to connect, specify them. For bastion-based access to database, `TARGETDATABASE_CONNECTIONDETAILS_BASTIONDETAILS_IP` specifies the IP address of the bastion host.

Property	Description
Syntax	`TARGETDATABASE_CONNECTIONDETAILS_BASTIONDETAILS_IP = `*`ip address`*
Default value	There is no default value

TARGETDATABASE_CONNECTIONDETAILS_BASTIONDETAILS_PORT

The `TARGETDATABASE_CONNECTIONDETAILS` parameters specify connection details for the target OCI database. These are optional for Autonomous Database; however if an HTTP proxy is required to connect, specify them. For bastion-based access to database, `TARGETDATABASE_CONNECTIONDETAILS_BASTIONDETAILS_PORT` specifies the port number of the bastion host.

Property	Description
Syntax	`TARGETDATABASE_CONNECTIONDETAILS_BASTIONDETAILS_PORT =` *`port_number`*
Default value	There is no default value

TARGETDATABASE_CONNECTIONDETAILS_BASTIONDETAILS_REMOTEHOSTIP

The `TARGETDATABASE_CONNECTIONDETAILS` parameters specify connection details for the target OCI database. These are optional for Autonomous Database; however if an HTTP proxy is required to connect, specify them. For bastion-based access to database, `TARGETDATABASE_CONNECTIONDETAILS_BASTIONDETAILS_REMOTEHOSTIP` specifies the remote host IP address to access from the bastion.

Property	Description
Syntax	`TARGETDATABASE_CONNECTIONDETAILS_BASTIONDETAILS_REMOTEHOSTIP =` *`ip address`*
Default value	There is no default value

TARGETDATABASE_CONNECTIONDETAILS_BASTIONDETAILS_USERNAME

The `TARGETDATABASE_CONNECTIONDETAILS` parameters specify connection details for the target OCI database. These are optional for Autonomous Database; however if an HTTP proxy is required to connect, specify them. For bastion-based access to database, `TARGETDATABASE_CONNECTIONDETAILS_BASTIONDETAILS_USERNAME` specifies the user name to access the bastion.

Property	Description
Syntax	`TARGETDATABASE_CONNECTIONDETAILS_BASTIONDETAILS_USERNAME =` *`username`*
Default value	There is no default value

TARGETDATABASE_CONNECTIONDETAILS_HOST

The `TARGETDATABASE_CONNECTIONDETAILS` parameters specify connection details for the target OCI database. These are optional for Autonomous Database; however if an HTTP proxy is required to connect, specify them. `TARGETDATABASE_CONNECTIONDETAILS_HOST` specifies the listener host name or IP address.

Property	Description
Syntax	`TARGETDATABASE_CONNECTIONDETAILS_HOST =` *`hostname_or_ip`*
Default value	There is no default value

TARGETDATABASE_CONNECTIONDETAILS_PORT

The `TARGETDATABASE_CONNECTIONDETAILS` parameters specify connection details for the target OCI database. These are optional for Autonomous Database; however if an HTTP proxy is required to connect, specify them. `TARGETDATABASE_CONNECTIONDETAILS_PORT` specifies the listener port number.

Property	Description
Syntax	`TARGETDATABASE_CONNECTIONDETAILS_PORT =` *`listener port number`*
Default value	There is no default value

TARGETDATABASE_CONNECTIONDETAILS_PROXYDETAILS_HOSTNAME

The `TARGETDATABASE_CONNECTIONDETAILS` parameters specify connection details for the target OCI database. For connecting to the target database through an HTTPS proxy, `TARGETDATABASE_CONNECTIONDETAILS_PROXYDETAILS_HOSTNAME` specifies the proxy host name.

Property	Description
Syntax	`TARGETDATABASE_CONNECTIONDETAILS_PROXYDETAILS_HOSTNAME =` *`proxy host name`*
Default value	There is no default value

TARGETDATABASE_CONNECTIONDETAILS_PROXYDETAILS_PORT

The `TARGETDATABASE_CONNECTIONDETAILS` parameters specify connection details for the target OCI database. For connecting to the source database through an HTTPS proxy, TARGETDATABASE_CONNECTIONDETAILS_PROXYDETAILS_PORT specifies the HTTP proxy port number.

Property	Description
Syntax	`TARGETDATABASE_CONNECTIONDETAILS_PROXYDETAILS_PORT = ` *`proxy port number`*
Default value	There is no default value

TARGETDATABASE_CONNECTIONDETAILS_SERVICENAME

The TARGETDATABASE_CONNECTIONDETAILS parameters specify connection details for the target OCI database. These are optional for Autonomous Database; however if an HTTP proxy is required to connect, specify them.
TARGETDATABASE_CONNECTIONDETAILS_SERVICENAME specifies the fully qualified service name.

Property	Description
Syntax	`TARGETDATABASE_CONNECTIONDETAILS_SERVICENAME = ` *`service name`*
Default value	There is no default value

TARGETDATABASE_CONNECTIONDETAILS_TLSDETAILS_CREDENTIALSLOCATION

The TARGETDATABASE_CONNECTIONDETAILS parameters specify connection details for the target OCI database. These are optional for Autonomous Database; however if an HTTP proxy is required to connect, specify them. For a TLS connection, TARGETDATABASE_CONNECTIONDETAILS_TLSDETAILS_CREDENTIALSLOCATION specifies the directory containing client credentials (wallet, keystore, trustfile, etc.) Not required if using TCP.

Property	Description
Syntax	`TARGETDATABASE_CONNECTIONDETAILS_TLSDETAILS_CREDENTIALSLOCATION = ` *`directory`*
Default value	There is no default value

TARGETDATABASE_CONNECTIONDETAILS_TLSDETAILS_DISTINGUISHEDNAME

The TARGETDATABASE_CONNECTIONDETAILS parameters specify connection details for the target OCI database. These are optional for Autonomous Database; however if an HTTP proxy is required to connect, specify them. For a TLS connection, TARGETDATABASE_CONNECTIONDETAILS_TLSDETAILS_DISTINGUISHEDNAME specifies the distinguished name (DN) of the database server (SSL_SERVER_CERT_DN). Not required if using TCP.

Property	Description
Syntax	TARGETDATABASE_CONNECTIONDETAILS_TLSDETAILS_DISTINGUISHEDNAME = *distinguished_name*
Default value	There is no default value

TARGETDATABASE_GGADMINUSERNAME

The TARGETDATABASE parameters specify connection details for the target OCI database. For online logical migrations, TARGETDATABASE_GGADMINUSERNAME specifies the GoldenGate administrator user name.

Property	Description
Syntax	TARGETDATABASE_GGADMINUSERNAME = *gg_admin_username*
Default value	There is no default value

TARGETDATABASE_OCID

The TARGETDATABASE parameters specify connection details for the target OCI database. TARGETDATABASE_OCID specifies the Oracle Cloud resource identifier.

Property	Description
Syntax	TARGETDATABASE_OCID = *ocid*
Default value	There is no default value

H
Zero Downtime Migration ZDMCLI Command Reference

ZDMCLI is the command line interface for Zero Downtime Migration migration operations.

To run ZDMCLI commands, go to the `/bin` directory in the Zero Downtime Migration software home and enter one of the commands listed in the topics below. For example:

```
zdmuser> $ZDM_HOME/bin/zdmcli migrate database
```

To list help pages for any ZDMCLI command use the `-help` option. The following command lists the help for all of the ZDMCLI commands.

```
zdmuser> $ZDM_HOME/bin/zdmcli -help
```

To get the current release information for your Zero Downtime Migration software, run ZDMCLI with the `-build` option.

```
zdmuser> $ZDM_HOME/bin/zdmcli -build
```

The following topics describe the Zero Downtime Migration ZDMCLI command usage and options.

abort job

Terminates the specified job, if running.

Syntax

```
$ZDM_HOME/bin/zdmcli abort job
  -jobid job_id
```

Options

Table H-1 ZDMCLI abort job Options

Option	Description
-jobid *job_id*	Unique job ID value (integer) for the scheduled job. The job ID is assigned when the job is scheduled.

add imagetype

Configures a new image type of the specified name and its associated user actions.

Syntax

```
$ZDM_HOME/bin/zdmcli add imagetype
  -imagetype image_type
  -basetype CUSTOM_PLUGIN
  [-useractions user_action_list]
```

Options

Table H-2 ZDMCLI add imagetype Options

Option	Description
`-imagetype` *`image_type`*	Name of the image type to be created
`-basetype CUSTOM_PLUGIN`	The base image type for which the image type is created. Note that `CUSTOM_PLUGIN` is the only valid value for this mandatory argument.
`-useractions` *`user_action_list`*	Comma-separated list of user action names

add useraction

Configures a new user action of the specified name with its associated script and action file.

Syntax

```
$ZDM_HOME/bin/zdmcli add useraction
  -useraction user_action_name
  -actionscript script_name
  [-actionfile file_name]
  {-pre | -post}
  -optype MIGRATE_DATABASE
  [-phase operation_phase]
  [-onerror {ABORT | CONTINUE}]
  [-runscope
     {ONENODE |
      ALLNODES |
      AUTO}]
```

Options

Table H-3 ZDMCLI add useraction Options

Option	Description
`-useraction` *`user_action_name`*	Name of the user action
`-actionscript` *`script_name`*	Script file to be run
`-actionfile` *`file_name`*	File associated with and needed by the user action

Table H-3 (Cont.) ZDMCLI add useraction Options

Option	Description
`-pre`	Runs the user action before the operation
`-post`	Runs the user action after the operation
`-optype MIGRATE_DATABASE`	Defines the operation for which the user action is configured as `MIGRATE_DATABASE`.
`-phase` *`phase_of_operation`*	Migration operation phase for which the user action is configured
`-onerror {ABORT | CONTINUE}`	The response if the user action encounters an error during execution
`-runscope {ONENODE | ALLNODES | AUTO}`	The servers on which the user action is run. Specify *AUTO* to choose the run scope based on the other command options.

migrate database

Performs a migration of a database to the Oracle Cloud.

Syntax

```
$ZDM_HOME/bin/zdmcli migrate database
  -rsp zdm_template_path
  -sourcenode source_host_name
 [{-srcroot |
    -srccred cred_name |
    -srcuser user_name |
    {-srcsudouser sudo_user_name -srcsudopath sudo_binary_path} |
    {-srcauth plugin_name
       [-srcarg1 name1:value1
          [-srcarg2 name2:value2...]]}}]
 {-sourcedb db_name |
   -sourcesid source_oracle_sid}
  -targetnode target_host_name
{-tgtroot |
   -tgtcred cred_name |
   -tgtuser user_name |
   {-tgtsudouser sudo_user_name -tgtsudopath sudo_binary_path} |
   {-tgtauth plugin_name
      [-tgtarg1 name1:value1
         [-tgtarg2 name2:value2...]]}}
  [-eval]
  -backupuser user_name
  [-targethome target_home]
  [-imagetype]
  [-tdekeystorepasswd]
  [-tdemasterkey]
  [-sourcesyswallet sys_wallet_path]
  [-osswallet oss_wallet_path]
  [-tdekeystorewallet tde_wallet_path]
```

```
[-useractiondata user_action_data]
[-schedule {timer_value | NOW}]
[-pauseafter phase]
[-stopafter phase]
[-listphases]
[-ignoremissingpatches patch_name [,patch_name...]]
[-ignore {ALL | WARNING | PATCH_CHECK}]
[-advisor | [-ignoreadvisor | -skipadvisor]]
```

Options

Table H-4 ZDMCLI migrate database Options

Option	Description
`-sourcedb` *`db_name`*	Name of the source database you want to migrate
`-sourcenode` *`source_host_name`*	Host on which the source database is running
`-targetnode` *`target_host_name`*	Target server to which the source database is migrated
`-targethome` *`target_home`*	Location of the target database `ORACLE_HOME`
`-imagetype` *`image_type`*	Name of the user action imagetype
`-useractiondata` *`user_action_data`*	Value to be passed to *useractiondata* parameter of the user action script
`-rsp` *`zdm_template_path`*	Location of the Zero Downtime Migration response file
`-sourcesid` *`source_oracle_sid`*	`ORACLE_SID` of the source single instance database without Grid Infrastructure
`-eval`	Evaluate the migration job without actually running the `migrate database` command against the source and target
`-backupuser` *`user_name`*	Name of the user allowed to back up or restore the database
`-srcroot`	Directs Zero Downtime Migration to use `root` credentials to access the source database server
`-srccred` *`cred_name`*	Credential name with which to associate the user name and password credentials to access the source database server
`-srcuser` *`user_name`*	Name of the privileged user performing operations on the source database server
`-srcsudouser` *`user_name`*	Perform super user operations as `sudo` user name on the source database server
`-srcsudopath` *`sudo_binary_path`*	Location of `sudo` binary on the source database server

Table H-4 (Cont.) ZDMCLI migrate database Options

Option	Description

`-srcauth` *`plug-in_name`* `[`*`plug-in_args`*`]`	Use the `zdmauth` authentication plug-in to access the source database server, and enter the following arguments:
	
	`-srcarg1`
	`user:`*`source_database_server_login_user_name`*
	`-srcarg2`
	`identity_file:`*`ZDM_installed_user_private_key_file_location`*
	`-srcarg3`
	`sudo_location:`*`sudo_location`*
	
	If you don't specify the sudo location the default (/usr/bin/sudo) is used by ZDM.

`-tgtroot`	Use `root` credentials to access the target database server
`-tgtcred` *`cred_name`*	Credential name with which to associate the user name and password credentials to access the target database server
`-tgtuser` *`user_name`*	Name of the user performing operations on the target database server
`-tgtsudouser` *`user_name`*	Perform super user operations as `sudo` user name on the target database server
`-tgtsudopath` *`sudo_binary_path`*	Location of `sudo` binary on the target database server

`-tgtauth` *`plugin_name`* `[`*`plugin_args`*`]`	Use the `zdmauth` authentication plug-in to access the target database server, and enter the following arguments:
	
	`-tgtarg1`
	`user:`*`target_database_server_login_user_name`*
	`-tgtarg2`
	`identity_file:`*`ZDM_installed_user_private_key_file_location`*
	`-tgtarg3`
	`sudo_location:`*`sudo_location`*
	
	If you don't specify the sudo location the default (/usr/bin/sudo) is used by ZDM.

`-tdekeystorepasswd`	Transparent Data Encryption (TDE) keystore password, required for password-based keystore or wallet
`-tdemasterkey`	Transparent Data Encryption (TDE) master encryption key

Table H-4 (Cont.) ZDMCLI migrate database Options

Option	Description
`-schedule` *`timer_value`*	Scheduled time to execute the operation, in ISO-8601 format. For example: 2016-12-21T19:13:17+05
`-pauseafter` *`phase`*	Pause the job after running the specified phase
`-ignoremissingpatches`	Proceed with the migration even though the specified patches, which are present in the source path or working copy, might be missing from the destination path or working copy
`-ignore {ALL | WARNING | PATCH_CHECK}`	Ignore all checks or specific type of checks
`-listphases`	List the phases for this operation
`-sourcesyswallet` *`sys_wallet_path`*	Full path for the auto-login wallet file on the Zero Downtime Migration host containing the `SYS` password of the source database
`-osswallet` *`oss_wallet_path`*	Full path for the auto-login wallet file on the Zero Downtime Migration host containing the credential for the Object Storage Service (OSS) backup user
`-tdekeystorewallet` *`tde_wallet_path`*	Full path for the auto-login wallet file on the Zero Downtime Migration host containing the Transparent Data Encryption (TDE) keystore password
`-advisor`	Only runs the minimum phases required for exclusively running Cloud Premigration Advisor Tool (CPAT) on the migration
`-ignoreadvisor`	Ignore Cloud Premigration Advisor Tool (CPAT) errors
`-skipadvisor`	When this option is included, Cloud Premigration Advisor Tool (CPAT) does not run when `MIGRATE DATABASE` is executed
`-stopafter` *`phase`*	Truncates the workflow by the specified phase, and upon completion of the specified phase, the migration job is marked with status completed. There is no option to resume the job beyond this phase. For example, if you intend to stop the workflow after Data Guard setup, then specifying `–stopafter ZDM_CONFIGURE_DG_SRC` stops the job at this phase and the job is marked completed once the `ZDM_CONFIGURE_DG_SRC` is completed successfully.

Examples

ZDMCLI `migrate database` options for an Autonomous Database migration:

```
zdmuser> $ZDM_HOME/bin/zdmcli migrate database -rsp file_path -sourcenode
host
 -srcauth zdmauth -srcarg1 user:username -srcarg2 identity_file:ssh_key_path
 -srcarg3 sudo_location:sudo_path -eval [-advisor [-ignoreadvisor] | -
skipadvisor]]
```

ZDMCLI `migrate database` options for a co-managed database migration:

```
zdmuser> $ZDM_HOME/bin/zdmcli migrate database -rsp file_path -sourcenode
host
 -srcauth zdmauth -srcarg1 user:username -srcarg2 identity_file:ssh_key_path
 -srcarg3 sudo_location:sudo_path -targetnode host -tgtauth zdmauth
 -tgtarg1 user:username -tgtarg2 identity_file:ssh_key_path
 -tgtarg3 sudo_location:sudo_path -eval [-advisor [-ignoreadvisor] | -
skipadvisor]]
```

modify useraction

Modifies the configuration of the user action with the specified name.

Syntax

```
$ZDM_HOME/bin/zdmcli modify useraction
  -useraction user_action_name
  [-actionscript script_name]
  [-actionfile file_name]
  [-pre | -post]
  [-optype MIGRATE_DATABASE]
  [-phase phase]
  [-onerror {ABORT | CONTINUE}]
  [-runscope
     {ONENODE |
      ALLNODES |
      AUTO}]
```

Options

Table H-5 ZDMCLI modify useraction Options

Option	Description
`-useraction` *`user_action_name`*	Name of the user action
`-actionscript` *`script_name`*	Script file to be run
`-actionfile` *`file_name`*	Accompanying file needed by the user action
`-pre`	Runs the user action before the operation
`-post`	Runs the user action after the operation

Table H-5 (Cont.) ZDMCLI modify useraction Options

Option	Description
`-optype MIGRATE_DATABASE`	Defines the operation for which the user action is configured as `MIGRATE_DATABASE`
`-onerror {ABORT | CONTINUE}`	Defines whether to stop or continue running if an error occurs while the user action is running
`-runscope {ONENODE | ALLNODES | AUTO}`	The servers where the user action will be run. Specify *AUTO* to choose the run scope based on the other command options.

query job

Gets the current status of scheduled migration jobs.

Syntax

```
$ZDM_HOME/bin/zdmcli query job
  [-jobid job_id
     [-jobtype]]
  [-sourcenode source_host_name
     [-sourcedb db_name |
     -sourcesid sid]]
  [-targetnode target_host_name]
  [-latest]
  [-eval |
  -migrate]
  [-status
     {SCHEDULED |
      EXECUTING |
      UNKNOWN |
      TERMINATED |
      FAILED |
      SUCCEEDED |
      PAUSED |
      ABORTED}]
  [-phase]
  [-dbname database_name]
  [-since timer_value]
  [-upto timer_value]
  [-brief]
  [-statusonly]
```

Options

Table H-6 ZDMCLI query job Options

Option	Description
`-jobid` *`job_id`*	Unique job ID value (integer) for the scheduled migration job The job ID is assigned when the migration job is scheduled.
`-job_type`	Returns the type of the scheduled job
`-sourcenode` *`source_host_name`*	Server on which the source database is running
`-sourcedb` *`db_name`*	Name of the source database to be migrated
`-sourcesid` *`sid`*	The `ORACLE_SID` of the source single instance database without Grid Infrastructure
`-targetnode` *`target_host_name`*	Target server to which the database is migrated
`-latest`	Returns the most recent job that matches the given criteria
`-eval`	Returns evaluation jobs only
`-migrate`	Returns migration jobs only
`-status {SCHEDULED | EXECUTING | UNKNOWN | TERMINATED | FAILED | SUCCEEDED | PAUSED | ABORTED}`	Returns jobs that match the specified job status

`-phase`	Returns the status of the given phase. If the phase supplied by the user is invalid, the query returns an error.
	`./zdmcli query job -jobid 33 -phase ZDM_PREUSERACTIONS_TGT -statusonly`
	`# exmaple.com: Audit ID: 153`
	`# Job ID: 33`
	`# ZDM_PREUSERACTIONS_TGT:PENDING`

`-dbname` *`unique_db_name`*	Specifies the database `DB_UNIQUE_NAME` value
`-since` *`timer_value`*	Date from which to get the jobs, in ISO-8601 format. For example: 2016-12-21T19:13:17+05
`-upto` *`timer_value`*	Upper limit time to which to get the jobs, in ISO-8601 format. For example: 2016-12-21T19:13:17+05
`-brief`	Returns job details summary only
`-statusonly`	Returns only the job status and current phase name

Usage Notes

To identify if the current run is a resumption of a PAUSED migration job, a restart of a FAILED migration job, or a fresh start for a migration job, use the following options:

- `zdmcli query job -latest` gets the latest job without considering the `job_type`

- `zdmcli query job -latest -migrate` gets the latest non-evaluation migration job
- `zdmcli query job -latest -eval` get the latest evaluation migration job

query useraction

Displays the configuration of a user action.

Syntax

```
$ZDM_HOME/bin/zdmcli query useraction
    [-useraction user_action_name | -imagetype image_type
    [-optype MIGRATE_DATABASE]]
```

Options

Table H-7 ZDMCLI query useraction Options

Option	Description
`-useraction` *`user_action_name`*	Name of the user action
`-imagetype` *`image_type`*	Specify the image type name
`-optype MIGRATE_DATABASE`	Operation for which the user action is configured

resume job

Resumes a specified job that was paused.

Syntax

```
$ZDM_HOME/bin/zdmcli resume job
  -jobid job_id
  [-pauseafter pause_phase]
```

Options

Table H-8 ZDMCLI resume job Options

Option	Description
`-jobid` *`job_id`*	Unique job ID value (integer) for the scheduled job The job ID is assigned when the migration job is scheduled.
`-pauseafter` *`pause_phase`*	Pausees the migration job after running the specified phase

suspend job

Suspends the specified job if running. Executing `suspend job` stops the ongoing job at the current workflow phase and allows jobs to be resumed later.

Syntax

```
$ZDM_HOME/bin/zdmcli suspend job
  -jobid job_id
```

Options

Table H-9 ZDMCLI suspend job Options

Option	Description
`-jobid` *`job_id`*	Unique job ID value (integer) The job ID number is assigned when the migration job is scheduled.

Index